MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

AD-A151 936

IMPLEMENTATION OF A REAL-TIME,

INTERACTIVE, CONTINUOUS SPEECH

RECOGNITION SYSTEM

THESIS

Kathy R. Dixon
First Lieutenant, USAF

AFIT/GE/ENG/84D-26

DTIC
ELECTE
MAR 2 8 1985
S D
E

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

85   03   13   103

IMPLEMENTATION OF A REAL-TIME,

INTERACTIVE, CONTINUOUS SPEECH

RECOGNITION SYSTEM

THESIS

Kathy R. Dixon
First Lieutenant, USAF

AFIT/GE/ENG/84D-26

DTIC
ELECTE

E
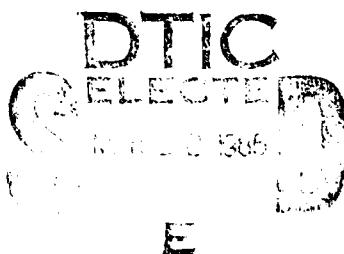
AFIT/GE/ENG/84D-26

IMPLEMENTATION OF A REAL-TIME, INTERACTIVE,

CONTINUOUS SPEECH RECOGNITION SYSTEM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Kathy R. Dixon
First Lieutenant, USAF

Graduate Electrical Engineering

December 1984

Preface

The purpose of this research was to develop a real-time, continuous speech recognition system. Since many computer algorithms existed in the AFIT Signal Processing Laboratory to characterize various aspects of human speech, there was a need to combine these programs into a viable system.

The system described in this paper provides an interactive means of continuous speech recognition by computer. The system is speaker-dependent and requires training with a 70-word vocabulary prior to word recognition.

Completing this research would have been impossible without the assistance of several people. I would like to thank my thesis advisor, Dr. Matthew Kabrisky, Professor of Electrical Engineering at the Air Force Institute of Technology for motivating this research. Also, I would like to thank MAJ Ken Castor for his encouragement and for reviewing this document. In addition, a note of gratitude to my many friends who have put up with me during the highs and lows of this research. I would especially like to thank my friend, CPT Michael Conrad (US Army) for his encouragement and enthusiasm, and for his invaluable typing assistance.

ii

Table of Contents

List of Figures

List of Tables

AFIT/GE/ENG/84D-26

Abstract

This thesis describes

A speech recognition system was designed and implemented to recognize continuous speech in a real time environment (after training). Several techniques were incorporated to characterize phonemes as vectors in space. Through the use of distance rules it was possible to characterize words by a phoneme representation, which could subsequently be used in word recognition. This approach to speech recognition offers several possibilities for future investigations such as varying the Minkowski distances and applying clustering techniques. The algorithm developed was modularized on a hierarchical basis and was user friendly. *Additional keywords: Software design, Computer applications, Computer programs*

IMPLEMENTATION OF A REAL-TIME, INTERACTIVE, CONTINUOUS SPEECH RECOGNITION SYSTEM

I. Introduction

Speech recognition by computer has seen dramatic gains in the past five years. Resulting primarily from increased computer processing power, efficient recognition algorithms can perform near real time. However, recognition systems using these algorithms are constrained to isolated word recognition and have limited vocabularies of between twenty and one hundred words. The major emphasis of present speech research is the development of an algorithm capable of recognizing natural, continuous speech (6:69).

Background

The goal of speech research is to determine a decoding scheme similar to that of the human brain. Many continuous speech recognition algorithms exist, yet none has a significant word recognition rate. There are three primary reasons for this, all of which are a consequence of our lack of understanding of how the human brain deals with the complex operation of speech recognition. First, the speech signal is highly encoded in the brain, and one must acquire an understanding of how such characteristics as intonation, articulation, semantics, and syntax are encoded before one

1

```
        SAMGEN   Rev.   2.10   7/29/82 at 10:49    Filename: SAMCONFIG3.SR

;Answers you gave in the SAMGEN dialog are shown in comment lines.
;Your inputs are immediately preceded by a colon (:) and appear
;in the same order as you gave them to SAMGEN.

; Target operating system type :MRD
; Number of DO/DAC 4300 chassis  configured: 0
; Fatal error handler name    : -1
; Fatal error handler mailbox: -1

        DCB.X    SAMCO  100 -1    -1


; Number of Analog Subsystem  :1



; A/D Con. #1  Device Code :21  Mode :AD    Fortran ID =  IDS21

; External interrupt handler specified : <NONE>
; Number of pages in Data Channel area :16
; Specifying a starting address for Data Channel area :Y
; Data Channel starting address : IBUFF

        DCB.M  DBS21   D.IDF+D.INF+D.DCH    21
        DCB.I  DTS21   SAINI      16.   IBUFF
        DCB?C    -1     -1    DS821
        DCT.M  DTS21   000377  INTSA   DSS21

        DCB.N    921  D.FIF     21     00     AD
        DCB.S  DBS21    0     AD.IS  AD.IN  SAIRT
        DCB.A



; D/A Con. #1  Device Code :23  Mode :BD    Fortran ID =  IDS23

; External interrupt handler specified : <NONE>
; Number of pages in Data Channel area : 1
; Specifying a starting address for Data Channel area :Y
; Data Channel starting address : IBUFO

        DCB.M  DBS23   D.IDF+D.INF+D.DCH    23
        DCB.I  DTS23   SAINI       1.   IBUFO
        DCB?C    -1     -1    DSS23
        DCT.M  DTS23   000377  INTSA   DSS23

        DCB.N    923  D.FIF     23     00     BD
        DCB.S  DBS23    0     BD.IS  BD.IN  SAIRT
        DCB.A

        DCB.E

.End of SAMGEN configuration file.
```

Figure 3.   SAM Configuration File

$$IDATA2 = (sample\ length)*800, \qquad (2)$$

where sample length is the duration in seconds of each audio input. Variable, IDATA3, specifies the locations where conversion values are placed. In FORTRAN, IDATA3 is an integer array placed in a labeled common block. The block label is the same as that given in the SAMGEN configuration files.

Octal Value	Function
00000K	pulse clock
20000K	DCH clock
40000K	internal clock
60000K	external clock
0-1700K	start channel 0-15
0-17K	final channel 0-15

Table 3. Octal Values for Bit Setting of IDATA1 (1:69)

Configuration files, produced by an interactive dialog with the program SAMGEN, define operating system hardware and operation nodes. For example, SAMCONFIG3 has the source file shown in Figure 3.

The twelve bits in the machine word produce 2^{12}, or 4096 different conversion values. Over a range of 10 volts, this means that each value increment represents .024 volts. Using a one-to-one correspondence between sampled values and integer values, the real value for voltage can be determined by the equation

$$\text{Real Value} = \text{Float(Integer Value)}/32768.*5. \qquad (1)$$

Variables (IDATAx words) are passed to the software routines to represent channel use numbers, conversion count, clock source and storage locations for conversion values. IDATA1, occupying one machine word, represents clock source and channel use numbers. There are four clock sources available to the device: pulse, DCH, internal and external. Though all four can be used for A/D conversion, only the external clock is accessed. Channel numbers are given as starting and ending channels. If both are the same only one channel is specified. The bit values of IDATA1 can be set using the the octal values in Table 3. Since an external clock and all sixteen channels are used, IDATA1 equals 61700K.

IDATA2, also occupying one machine word, specifies the total conversion count or the number of data samples. Thus, using an 800 Hz external clock, IDATA2 is computed by

A/D Conversion

A/D conversion is performed by making use of the model 4331 Eclipse analog device and independent software interfaces. Though both A/D and D/A capabilities are present, only the former is addressed.

The Eclipse A/D conversion device operates using device op-codes and conversion data buffers. Organized around a single 12-bit converter and two multiplexers, the device can accept up to 16 different input signals with voltage levels in a $\pm$ 5V range. The conversion values are stored as a machine word with the bit assignments shown in Table 2.

Bit	Assigned Value
0	sign
1-11	storage values
12-15	zero

Table 2. Bit Assignment for Control Words

f_0 (Hz)	Bandwidth(Hz)	Approximate Band Coverage	
		f_1	f_h
260	130	203	333
390	130	330	460
520	130	459	589
650	130	588	718
780	130	718	848
910	140	843	983
1060	160	983	1143
1220	180	1133	1313
1400	200	1303	1503
1600	220	1494	1713
1820	250	1699	1949
2070	300	1925	2225
2370	340	2206	2546
3035	1030	2563	3593
4272	1445	3610	5055
5997	2005	5077	7083

Table 1.   ASA-16 Analysis Bands (5:12)

11

Constant voltage level inputs to the ASA-16 chip are maintained by the automatic gain control circuit. Designed to accept 20 mV to 20 V over a 60 dB dynamic range, the AGC outputs voltages of 1 V to 4V, peak to peak.

To define band energies, the ASA-16 chip uses sixteen channels, each composed of a second-order bandpass filter, half-wave rectifier and a low-pass filter. Each channel is sequentially accessed using a sampled-and-held multiplexer. Table 1 shows the center frequencies and bands associated with each filter. A TTL crystal controlled 1 MHz clock is used for timing. Since the processor is built on a single board, both the chip and the board are powered by a common ± 10 V power supply.

Buffers are used between the chip outputs and the 4331 Eclipse A/D converter inputs, to correct DC offset and satisfy the 200kohm termination resistance that each bandpass filter requires (5:8-41).
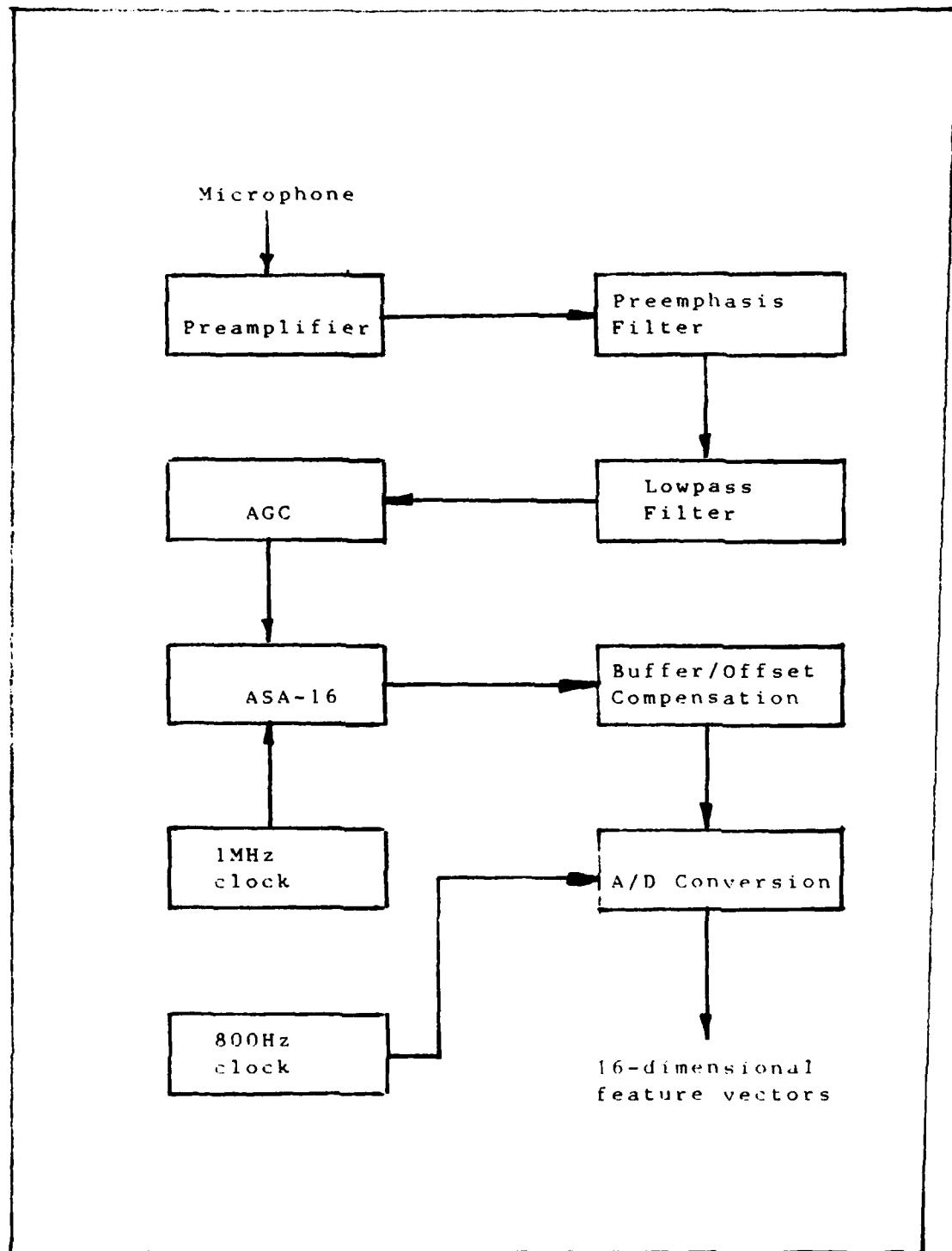
Figure 2. Acoustic Processor Designed by Hussain

II. Speech Processing

The word recognition algorithm developed in this thesis
is based on the output of the acoustic processor designed by
Ajmal Hussain (5). Built on a single board, the processor
conditions the analog speech prior to analog/digital
conversion. This chapter discusses the transformation of
speech from an analog signal to a set of feature vectors
used in pattern recognition.

Processor

Prior to A/D conversion, the audio signal is precondi-
tioned to preserve frequency components and limit signal
voltage levels. The primary component of this conditioning
process is the ASA-16 spectrum analyzer chip. Figure 2
illustrates what happens to the speech after entering the
system via a dynamic microphone and preamplification.

A preemphasis filter and an active low-pass filter band
limit the audio signal from 200 Hz to 7000 Hz, as required
by the ASA-16 chip. The preemphasis filter, with 6
dB/octave gain above 500 Hz, preserves the signal's high
frequency · components. The active low-pass filter then
defines the signal's upper frequency at 7000Hz (cutoff
frequency). A 17 dB passband gain provides proper voltage
levels for the automatic gain control circuit which follows.

8
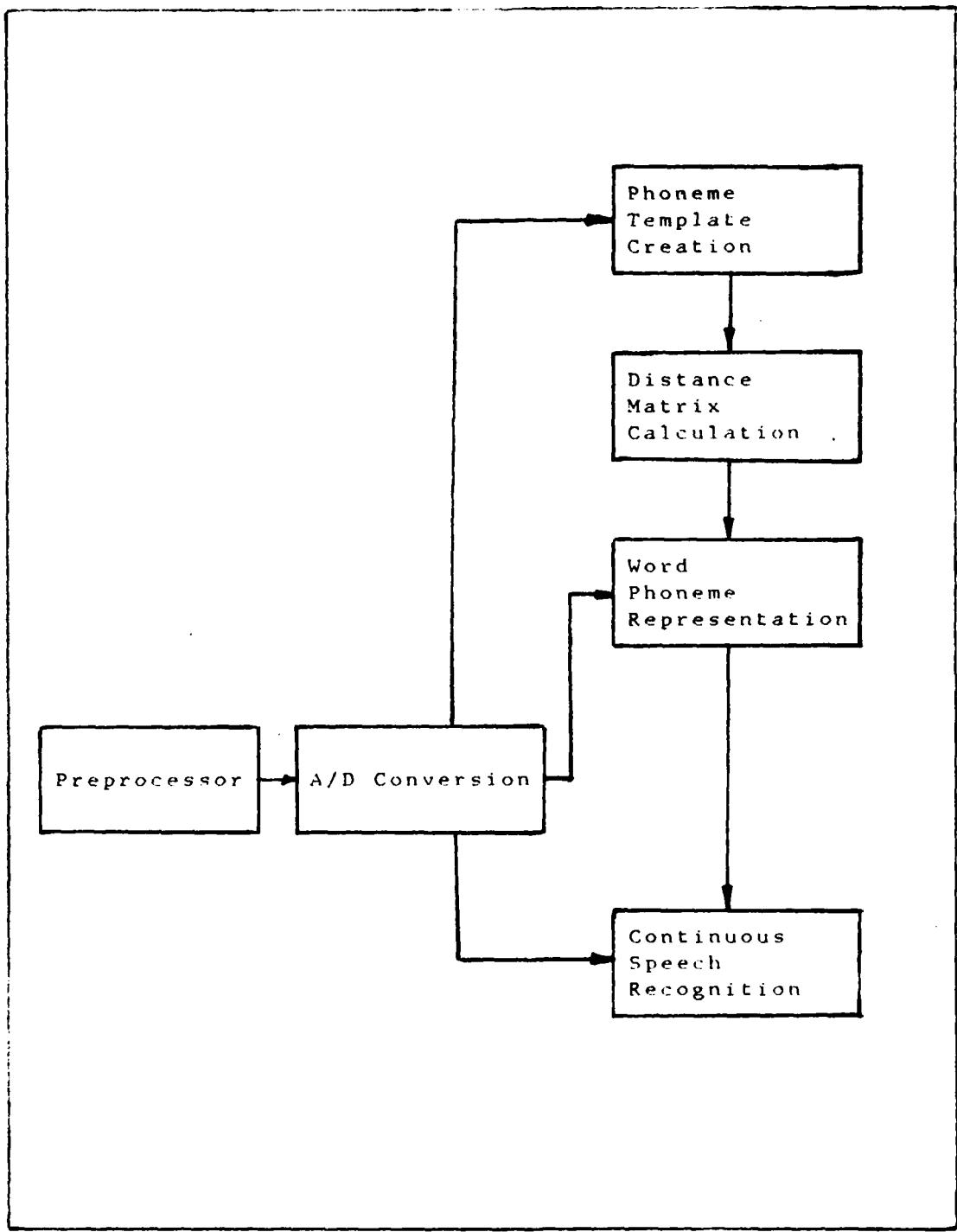
Figure 1. Speech Recognition System

two components of the machine: first, the acoustic processor (developed by Hussain) which provides a vector representation of the speech signal, and the recognition algorithm for determining a phoneme representation for each speaker, establishing templates for each word in the vocabulary (Appendix A), and finding words in a continuous speech phrase. More detailed outlines of these components are described in subsequent chapters.

Materials and Equipment

The following materials and equipment will be used:

1. Data General Eclipse S/250 Computer,

2. Filter Bank - Acoustic Processor.

Sequence of Presentation

The information contained in this report is presented in the following manner. Chapter 2 provides an explaination of speech processing both the digitization and frequency sampling of the speech signal. Chapter 3 describes the specifics of word recognition. Next, Chapter 4 is a synopsis of algorithm design. Finally, recognition results and appropriate conclusions and recommendations are presented in chapters 5 and 6, respectively.

front ends have been developed using fast Fourier transforms and filter banks. Spectrograms of speech waveforms have also provided information as to the possible waveform characterization using energy components. From these investigations, Seelandt (10) developed a universal feature set representing phonemes found in digitized speech. Then, Seelandt, Montgomery (7), and Hussain (5), suceeded in designing algorithms to recognize both isolated and connected speech by use of distance rules and template matching.

The purpose of this thesis is to combine several techniques developed at the AFIT Signal Processing Laboratory with additional modifications to produce a viable speech recognition system. A complete system, one in which analog speech is input and the recognized words are output , did not previously exist.

Two primary considerations in the design of this system were modularization and user friendliness. Modularity provides easy modification to system components, and user friendliness provides an operational machine which can be demonstrated by someone who may not be a computer expert.

General Approach

The speech recognition system developed from this research is described in Figure 1. Basically, there are

5

This means that phonemes at word boundaries are transitional and do not reveal much word information.

Summary of Current Knowledge

Research reports show extensive studies in the area of continuous speech recognition (4). Two methods currently being used for continuous speech recognition are segmentation and whole-word template matching. Segmentation is a means by which the acoustic signal is divided into unique sound units. After the word has been segmented, an attempt is made to match the sequence of units to a particular vocabulary word. The problem, though, is that some phonemes are lost in the process of chopping the signal into time slices. The other method, whole-word template matching, is means of matching whole word phoneme representations to the strings of phonemes produced by an acoustic processor. Problems inherent in this technique due to the variability of the word's duration when spoken. In the effort to find a solution some success has been achieved through the characterization of a word by phonemes from the body of the word. Thus, eliminating transition problems (4:574).

The development of a speech recognition machine has been an important area of research at the Air Force Institute of Technology for several years. Several acoustic

features such as formant frequencies, correlation coefficients, or linear predictive coefficients for pattern matching. Once the acoust ʌ signal is encoded, templates are constructed for each word. These templates are subsequently matched to encoded speech for word hypothesis. Although proven effective for isolated speech, this method is difficult to apply to continuous speech recognition which has undefined word boundaries and variable word durations. To remedy this situation, algorithms incorporate partial template matching techniques. Thus, the object of this research was the development of a continuous speech recognition system relying on a universal feature set and partial template matching.

Scope

The computer algorithm developed in this thesis was designed to recognize speaker dependent, continuous speech. Some considerations for acoustic processor error are made by calculating error statistics and incorporating them into the word determination phase of the algorithm.

Assumptions

This research was based on two assumptions. First, the output of the acoustic processor is reliable such that phoneme choices are consistent best guesses. Second, words can be defined by an incomplete phoneme representation.

3

can hope to perform artificial decoding Second, speech is a variable signal. Each time a speaker says a particular word, the phoneme representation of the word differs slightly. To the human ear, the variation is almost indistinguishable, but a machine must rely upon mathematical exactitudes. Therefore, any discrepancies are magnified in mecanical speech processing. Finally, although the brain somehow can distinguish the separations between words, these word boundaries are difficult for machines to find because the acoustic signal has no apparent pauses (4:570).

Fast, reliable man-machine communication is becoming a necessity as computers become integrated into today's society. Speech, the most natural form of human communication, seems appropriate in this application (6:64). Already, current testing onboard the Air Force's AFTI F-16, has proven that speech recognition is a definite aid to a pilot taxed to his physical limits by mechanical tasks.

Problem

The major problem in speech recognition lies not in the characterization of the acoustic signal, but in the determination of a decision scheme which uses these characteristics for word recognition. Several decision schemes have been developed which use acoustic signal

In the figure items worthy of particular attention are the device code number for A/D (IDS21), the number of pages in the data channel (16 blocks x 1024 words/block = 16384 words) and the data channel starting address IBUFF (area in common memory) (1:60-70).

To perform A/D conversion, the sequence of software commands is

```
EXTERNAL IDS21
EXTERNAL IDS23
COMMON/IBUFF/IDATA3(16384)
COMMON/IBUFO/IDATAO
DIMENSION IORBA(16)


CALL DSTRT(IER).

CALL DOIT[/W](IORBA,device-id,8,IDATA1,IDATA2,IDATA3,
             IER).
```

By way of explaination, the first few lines declare both the device identification numbers and the common areas. After that, DSTRT performs initialization, and DOIT requests the conversion operation. Errors are reported by IER values as shown in Table 4. Any additional error is reported by IORBA(14). If IORBA(14) does not equal 40000K after conversion, an external clock interrupt or a clock overrun/underrun has occurred (1:78-85). A complete listing of this procedure can be found in the ATOD.FR source code in Appendix C.

Value	Meaning	Value	Meaning
2179	No - LNK routine in DCB, invalid DCB. Often results from an invalid device-id, so check the device-ids. The first two characters are ID, the third either S, A, or O, and the last two are numbers (e.g., IDS21).	2192	Illegal conversion count -- more than 255 or less than 1 -- for an A/D converter mode in A2; DG/DAC only.
2180	No DCB identifier in IORB, invalid DCB. Same cause as 2179.	2193	Assembly language only. Attempt to move data channel map while IORB is locked. A task tried to change the map while a request was using the window.
2181	Not used. This error should not occur.	2194	Attempt to move data channel map to an address outside the window.
2184	No initializing routine for a device that needs initialization. Same cause as 2179.	2195	Illegal conversion count: less than 1 or more than the device allows.
2185	Output requested to a channel for an illegal device (e.g., output to an A/D converter).		
2186	Attempt to set up a locked IORB array. This can happen if a second DSAN/DSOR call uses the same IORB array argument before the original DSAN/DSOR completes.	2196	Interrupt occurred from 4222 without a strobe or latch change.
		2197	Assembly language only. Attempt to use data channel map while it is being initialized or moved.
2187	Unable to find free IORB block in IORB array. Can happen if the IORB array was DIMENSIONed too small. A multiple-operation call needs 8 elements + 8 elements per operation.	2198	Assembly language only. Data channel not initialized: use an RMAP call before issuing this mode A2 request.
2188	No DCB exists with specified device-id. Same cause as 2179.	2199	SAM panic code. SAM could not transmit (.IXMT) to the calling task on IORB array completion. SAM aborts the program unless you set up a fatal error handling RECeive task and gave its name to SAMGEN, as described in Chapter 5, "Initial Dialog".
2189	Attempt to use unsupported feature (e.g., mapped call in unmapped system).		
2190	Attempt to return bad buffer. Will never occur.	2200	External interrupt occurred on a stand-alone analog converter, aborting the request. This error returns from ISA calls only, not from DSAN/DSOR calls.
2191	An IDATAx argument gave an illegal clock setting for an A/D or D/A converter.		

Table 4. SAM Errors (1:82-84)

17

Vectors are sequentially ordered in the data array,
IDATA3. Each vector represents a time slice in which all
sixteen filters have been sampled. The sampling rate per
vector is 25 Hz, or once every 40 msec.

III. Word Recognition

As described in the introduction, word recognition in
continuous speech consists of making a phoneme template,
constructing word templates from the phoneme template, and
matching the word templates to an input phoneme string.
This chapter discusses the process in more detail.

Phoneme Representation

Since as early as 1947, phonemes have been used in word
recognition. Potter, Kopp, and Green (8) found that sounds
had unique spectrograms, and that people who they had
trained to visually read these spectrograms could identify
sounds in a word spectrogram. In other words, distinct
phoneme patterns were discernable in word spectrograms. In
1981, Seelandt (10) investigated digitized speech
spectrograms and the concept of phoneme patterns emerged.
By combining several time slices or vectors of digitized
speech, he produced a set of seventy phonemes. The concept
was carried a step further by Hussain (5) who produced
single-vector sixteen-dimensional phonemes. The method for
phoneme generation developed for this system compares and
averages these sixteen-dimensional vectors to produce a set
of less than seventy phonemes. The comparison and averaging
continue until a phoneme set is defined. This technique is
explained more clearly later in this chapter.

19

Distance Rules

Since they are suited to defining the spatial orientation of n-dimensional vectors, distance rules are key to vector metrics. An understanding of vector relationships allows one to reduce large vector sets and approximate vector strings by known vectors.

Many pattern recognition algorithms incorporate metrics based upon Minkowski distances. Table 5 presents some of these different calculations.

The Minkowski distance between two vectors is computed by the following procedure. The absolute difference between corresponding vector elements is calculated and then raised to the Minkowski power. Next, these values are summed and the resulting sum raised to (1/Minkowski power). The final value is the vector score, or the numerical relationship between the two vectors. This algorithm uses Minkowski 4 which empahasizes the effect of any large discrepancies between compared vectors (5).

Minkowski 1 $D_1 = \sum\limits_{i=1}^{M} | X_{i,m} - X_{i,n} |$

Minkowski 2 $D_2 = \left(\sum\limits_{i=1}^{M} (X_{i,m} - X_{i,n})^2 \right)^{\frac{1}{2}}$

Minkowski N $D_N = \left(\sum\limits_{i=1}^{M} (X_{i,m} - X_{i,n})^N \right)^{1/N}$

where   N = Minkowski distance   m = vector m
        M = Vector length   n = vector n
        i = vector element

Table 5. Minkowski Distances

Clustering

Rabiner, et al. (9) showed that speech vectors in space tend to cluster. Using this concept, his research group tried clustering analysis on phonemes, words, and speakers. The experiments were successful for isolated word recognition, but clustering became a computational nightmare in continuous speech recognition. As a result clustering was not emphasized in this research. However, some spatial comparisons are performed.

By visualizing the speech vectors as residing in n-dimensional space, one can imagine that some vectors lie closer together than others. Therefore, close vectors can be used to define regions of space, or clusters. As previously mentioned this algorithm redefines nearest vectors by averaging and weighting. Each vector has an initial weight of one, and those vectors which are closest to one another by Minkwoski distance are averaged together by the equation

$$X = \frac{(X * weight(m)) + (X * weight(n))}{weight(m) + weight(n)}. \qquad (3)$$

The highest-numbered vector is deleted and a new weight is assigned to the new vector by

$$weight(m) = weight(m) + weight(n). \qquad (4)$$

22

The resulting set of vectors represents clusters of similar sounds.

Phoneme Template

By incorporating the above mentioned concepts, a phoneme template is made for each speaker. This template, a set of energy normalized vectors, represents the most unique sounds produced by that speaker. The process is:

1. Vector normalization

2. Vector deletion

3. Vector comparison

4. Vector averaging.

Vector normalization consists of noise removal and energy normalization. Since background noise is present in all environments except an anechoic chamber, its removal from the audio signal allows for better characterization of the signal. The first vector in each speech file is considered to represent the average background noise. Therefore, by subtracting this vector from each of the remaining vectors, the noise is removed. To insure voltage consistency, a noise threshold voltage of 30.5 millivolts (corresponding to the average laboratory background noise) is set. Each element in a vector is checked to insure voltages above the noise threshold. If the value of the

23

element voltage falls below this threshold, the elements voltage is redefined as zero. Additionally, the vectors are energy normalized, by dividing each vector component by the vector's energy, calculated by

$$E_m = \sum_{i=1}^{16} (X_{i,m})^2 . \qquad (5)$$

These energy normalized vectors are now considered phonemes.

On the basis of energy thresholding, some vectors are omitted from the speech file. The energy threshold is determined by finding the average energy per vector over the test utterance. The equation is

$$E_{AVG} = (\sum_{m=1}^{500} E_m) / (number \ of \ vectors) . \qquad (6)$$

Once normalization and deletion have been performed, the phonemes are compared to one another by Minkowski 4 distance to determine each vectors nearest neighbor. Then, if the total number of vectors is greater than sixty-nine, the two nearest neighbors are averaged together using the

averaging routine described earlier (see Clustering, p.20), and replaced by a single vector which is then energy normalized. After all nearest neighbor averaging have been completed, a check is made to see if the number of remaining vectors is less than seventy (the maximum number of phonemes that the system can handle). If the number of phonemes is greater than that, the entire comparison/averaging process is repeated.

The final outcome of this procedure is a set of vectors, each of which represents a particular phoneme. This vector set is referred to hearafter as the phoneme template. Each phoneme is then assigned a number from one to at most sixty-nine.

Word Templates

After the phoneme set is established, it is used to develop a codebook of phoneme representations for each word in the given vocabulary (Appendix A). The process components are:

1. Normalization
2. Phoneme extraction
3. Word representation by phonemes
4. Vocabulary creation.

As is the case in the creation of phoneme templates, the

25

vectors in the digitized speech of each word are normalized.

At this point phoneme extraction takes place. Each vector in the phoneme template is compared to each vector in the vector string of the input speech. By calculating the Minkowski 4 distance between a template vector and a string vector, the closest (minimum distance) template vector is determined. Then the phoneme number associated with the template vector is placed in an array, and the process is repeated until all string vectors are represented by phoneme numbers.

Following the process of string representation, the array of phonemes is compressed to at most ten phonemes in the following manner. First, identical adjacent phonemes are represented as one phoneme. Then, the distances between adjacent phonemes are obtained from a distance matrix. Adjacent phonemes with distances between them of less than 11.0 (distance measures are normalized to 100) are compressed by deleting the highest numbered phonemes.

The final outcome of this process is a string of phonemes representative of a vocabulary word. Collectively, these strings represent the entire vocabulary. Table 6 shows the phoneme represenatations of several words.

Vocabulary Word	Phoneme Represenation								
ADVISE	23	34	66	38	52	61	25	37	14	0
BRAVO	40	9	10	52	41	36	0	31	0	0
CANCEL	17	16	27	0	29	0	37	0	0	0
ECHO	13	28	0	45	44	45	31	0	13	0
FIVE	13	0	61	62	26	25	27	36	0	0
FOUR	59	36	58	31	60	31	0	0	0	0
WEAPON	7	8	66	40	0	45	0	0	0	0
WEST	40	8	65	66	64	20	20	29	14	0
ZERO	20	1	24	6	5	6	40	31	0	0

Table 6.   Word Phoneme Representations

Word Spotting

Word spotting is the technique of identifying words in continuous speech. Using the phoneme template and word templates, speech recognition is accomplished by the following procedure:

1. Normalization

2. Phoneme extraction

3. Compression

4. Comparison.

The procedure for normalization, extraction, and compression of input speech is identical to that which is followed in the creation of word templates. Once the input speech is represented as a string of template phonemes, recognition is attempted.

The recognition scheme, based on the scheme described by Hussain (5), is as follows. After having undergone phoneme representation, the input string of phonemes is searched for two or more adjacent zeros. Since zero phonemes represent low energy noise phonemes, adjacent zero phonemes are considered word boundaries. When two word boundaries are found, the next step is to establish the number of phonemes between the boundaries. Less than three phonemes are considered noise, more than eight phonemes means that two words have been spoken.

One word identification is performed by finding the distance (Minkowski 4) between the the first word template phoneme and the first string phoneme. Subsequent word phonemes are then compared to the string phonemes until a match (best fit) has been found for each of the word phonemes. The distances are variations between word

phonemes and string phonemes, and are considered errors.
The errors are then summed to obtain the total error by

$$E_{TOTAL} = \sum_{i=1}^{M} (D_{m,n})^4 \qquad (7)$$

where D_4 is Minkowski 4, m is the string phoneme number, n
is the template phoneme number, and M is the number of
string phonemes. Error calculations are repeated twice by
shifting the word phoneme set one phoneme to the right, and
then one phoneme to the left of the original value. Then
the three resulting error values are averaged. After all of
the vocabulary words have been matched to the input string,
the word with the smallest average error is considered to be
the correctly recognized word.

For continuous recognition, in which two words exist
side by side, a slightly different error calculation is
performed. Phonemes are added to a buffer one at a time.
Then, after ten additions to the buffer, error statistics
are calculated. The algorithm decides at which point the
best word match occurs and chooses this word to be the first
word. A similar calculation is performed for the second
word, however, in this case the last phoneme of the first
word is used as the first phoneme of the second. Isolated

```
    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 1  0
 2 19  0
 3 38 25  0
 4 48 34 13  0
 5 40 30 28 22  0
 6 39 31 30 32 18  0
 7 50 41 39 36 27 20  0
 8 52 43 39 35 29 23 16  0
 9 57 50 44 41 37 33 27 12  0
10 61 54 49 47 43 37 31 22 12  0
11 61 55 51 50 45 40 36 32 23 11  0
12 44 36 33 39 34 23 28 28 24 23 25  0
13 38 39 51 62 53 45 59 64 68 71 69 51  0
14 22 33 54 64 53 48 59 64 68 71 69 53 41  0
15 21 39 58 67 57 55 62 66 68 70 68 57 59 18  0
16 28 16 39 48 39 31 40 42 44 47 48 31 39 38 44  0
17 25 19 44 54 41 36 45 49 52 55 55 38 36 36 44 14  0
18 27 18 42 52 41 34 44 48 52 55 56 36 26 36 47 17 10  0
19 24 13 37 47 35 33 48 51 57 61 60 42 29 33 42 16 15 14  0
20 33 40 57 64 56 51 60 64 68 70 69 55 43 16 32 44 43 41 41  0
21 33 23 41 51 37 32 48 53 59 64 63 44 23 39 51 23 27 18 15 44  0
22 30 18 38 47 33 30 42 44 52 56 56 40 38 40 47 14 18 24 13 42 15  0
23 25 13 40 49 37 34 47 51 56 61 60 43 35 39 44 16 10 15 10 41 23 15  0
24 28 15 37 47 38 34 47 51 57 61 61 41 22 36 47 18 17  9 11 42 14 22 14  0
25 39 35 40 47 34 24 28 32 30 30 29 17 48 46 47 26 36 39 38 40 42 34 38 39  0
26 52 47 50 41 39 33 34 29 28 20 25 39 57 56 60 40 44 48 50 58 53 44 50 52 16  0
27 31 29 40 49 35 26 39 39 38 40 40 21 36 39 46 17 26 23 26 42 30 24 28 27 17 33  0
28 37 27 26 39 29 26 31 31 32 37 42 19 42 46 53 26 33 27 33 49 34 31 39 31 24 38 19  0
29 54 60 70 73 69 68 71 74 76 77 76 70 65 39 42 64 62 63 62 22 69 65 62 64 61 66 61 66  0
30 28 21 15 29 21 29 33 39 41 46 48 28 41 43 46 28 34 30 27 45 30 28 31 26 31 43 27 15 60  0
31 35 29 24 39 32 22 32 34 40 45 47 26 33 43 53 27 22 24 31 30 21 29 41 20 14 46 17  0
32 36 29 16 22 27 17 26 26 32 36 38 23 48 51 59 34 40 37 35 54 28 39 37 34 31 36 33 19 48 14 17  0
33 37 30 33 31 24 31 30 30 34 38 42 39 59 49 46 31 36 39 39 50 39 30 36 38 29 37 31 32 56 27 40 34  0
34 43 33 39 39 36 33 42 38 38 42 45 49 49 62 54 51 34 39 44 38 59 47 35 38 44 38 44 40 44 58 39 50 49 16
35 46 34 30 29 27 34 32 32 31 36 42 47 42 44 61 60 43 48 48 62 41 44 47 42 44 44 37 66 32 41 39 20
36 37 27 39 46 37 27 34 37 38 39 39 20 35 44 54 28 31 23 30 46 29 33 32 26 23 34 17 14 66 23 12 28 41
37 31 29 42 52 42 39 47 51 54 59 59 39 13 36 49 25 29 19 21 41 14 29 27 13 30 51 25 30 63 30 21 37 43
38 45 35 30 29 24 23 23 22 26 31 37 27 59 50 59 39 45 43 46 59 48 42 46 45 33 37 37 21 68 24 29 22 23
39 52 47 43 44 38 33 31 22 16 19 24 15 61 56 48 52 53 18 22 27 32 70 40 40 33 34
40 41 31 27 34 32 20 26 27 30 33 39 17 48 51 57 32 36 32 39 52 40 37 40 37 24 35 25 13 67 20 18 16 38
41 37 52 50 52 44 39 37 36 28 24 19 26 63 63 44 49 51 52 59 63 58 52 55 56 21 16 37 43 72 47 49 38 44
42 69 65 60 59 53 50 46 32 20 18 21 32 78 77 73 54 63 66 69 76 72 64 69 70 33 28 44 50 80 58 57 50 45
43 52 47 43 44 36 32 29 19 17 22 27 25 61 60 58 36 47 47 51 61 54 44 51 52 18 25 27 32 70 40 40 33 29
44 37 27 23 31 21 16 26 29 30 36 41 19 41 47 54 26 34 29 29 50 30 28 32 29 24 37 20 13 68 17 17 14 33
45 26 16 27 37 27 29 38 42 48 53 54 34 28 29 38 16 24 17 14 42 16 17 20 13 34 47 23 20 62 19 14 23 30
46 26 17 34 44 36 31 41 45 51 55 56 36 23 34 45 21 22 12 17 40 16 26 21 10 36 49 25 22 61 22 14 29 38
47 27 12 16 24 25 29 39 37 43 47 50 32 42 42 44 24 29 28 25 43 30 24 26 25 42 44 27 23 61 20 25 22 26
48 37 22 17 13 19 29 39 34 43 48 50 35 51 52 54 33 41 39 34 39 38 36 35 39 44 42 25 40 20 29 24 26
49 50 38 23 16 16 30 32 30 39 46 50 41 64 60 68 50 56 54 49 67 53 49 52 50 49 51 51 36 74 23 37 24 31
50 43 37 30 32 33 29 34 33 28 28 31 16 54 56 59 33 43 40 43 56 47 47 41 39 38 34 36 33 70 31 29 20 38
51 80 76 71 68 69 62 58 49 33 28 28 42 90 89 89 63 79 76 80 88 84 76 80 81 44 39 56 61 90 69 68 60 57
52 86 83 78 76 71 68 65 50 38 34 35 50 95 93 89 71 81 82 86 92 89 82 86 87 49 43 61 68 95 76 75 67 63
53 91 88 85 82 74 67 63 52 47 44 47 61 100 97 93 80 87 98 91 96 91 86 90 92 60 53 71 77 98 82 79 71 69
54 59 55 52 53 46 41 39 31 29 29 19 23 67 67 66 43 54 54 58 67 62 54 58 59 22 19 33 40 76 49 48 42 40
55 25 16 39 49 38 39 49 53 58 63 62 43 23 31 42 19 19 12  8 39 16 19 13  6 40 52 29 33 61 28 23 36 38
56 24 16 40 50 40 38 51 55 61 65 64 46 29 33 43 21 13 11 12 40 21 23 12  9 42 51 31 36 61 31 28 38 41
57 25 20 43 53 42 37 50 54 59 63 62 43 22 33 44 22 15  8 17 41 16 26 18  7 40 52 28 34 63 31 29 39 41
58 45 39 34 40 38 32 38 36 33 38 44 14 49 53 58 31 40 37 41 54 44 40 43 41 22 32 20 70 31 29 25 41
59 41 33 26 33 34 29 37 36 39 37 31 16 44 50 57 31 38 33 35 52 37 36 38 33 23 28 24 21 68 29 23 16 41
60 44 33 22 26 39 31 37 34 38 40 40 29 54 57 62 42 42 40 45 41 38 39 40 27 71 24 26 15 42
61 61 58 57 59 50 45 45 43 39 36 28 36 67 64 63 52 56 58 59 63 60 56 59 61 30 18 44 52 71 54 52 46 49
62 43 40 41 39 37 28 36 40 41 43 36 28 44 44 37 40 41 43 38 40 42 21 21 30 37 62 38 33 36 36
63 31 22 40 50 37 32 42 47 52 56 57 37 24 27 37 22 26 18 18 37 15 24 25 13 36 48 24 25 58 27 20 39 34
64 30 22 24 31 22 26 29 30 32 36 41 27 46 43 42 25 30 30 31 49 39 28 32 32 23 24 21 37 14 27 22 16
65 47 43 42 44 35 35 33 33 33 31 33 34 37 60 55 51 39 43 46 46 55 50 42 47 49 22 28 28 33 60 39 43 37 20
66 46 38 53 49 48 41 38 39 26 21 22 24 31 68 66 63 42 52 53 57 66 61 52 57 58 23 22 32 37 72 43 47 39 32
```

Figure 6a.   Distance Matrix

Distance Matrix Creation

Seelandt (10) showed that it is more time efficient to look up distances between phonemes than to recalculate them. Therefore, a lower triangular matrix is formed by calculating the distances between the phonemes in the array PHON. These distance values are stored in the array DIS. The number of phonemes in the matrix is stored in position 2432. The code for this subroutine was modified from Hussain's code so that the distance between phonemes are raised to the power 0.25 and then multiplied by 100. The resulting matrix is shown in Figurse 6a and 6b. As is the case during phoneme template creation, these values are written to a file called DIST. Program TRAIN is capable of both writing to, and reading from, this file.

Figure 5. Phoneme Template Vector Elements

vectors could have the same nearest neighbor, it is necessary to find pairs of vectors which are each other's nearest neighbors. This requires a considerable amount of time, but it is important for accurate analysis of vector metrics. After a nearest neighbor pair is found, the vector elements of the lowest-numbered vector are recalculated using Equation 3 and energy normalized. The weight of each vector is recalculated by Equation 4, and the weight of the highest-numbered vector is set to zero. The process is repeated until all nearest neighbor pairs have been found. These vectors are then compressed by the subroutine CMPRS1, and the number of remaining vectors is examined. If this number is greater than sixty-nine, new nearest neighbors are determined and the averaging process is repeated. This entire operation process can take anywhere from five to ten minutes.

After the correct number of vectors has been obtained, the voltage values are stored in the array PHON and passed to TRAIN, which has options for writing them to a file (PHONE) and reading them from a file (PHONE). (Figure 5 depicts the voltage elements for each vector in the phoneme template file. The vectors are horizontally arranged such that phoneme 1 is positioned (1,1), (1,2), (1,3),....,(1,16)).

Figure 4.   Energy in Data Vectors

In the COMPARE subroutine, the distances between all
pairs of vector's are determined by Minkowski 4 distance
calculations.   Each vector's nearest neighbor is determined
by Hussain's algorithm. This concept is carried a step
further by the introduction of vector weighting. Since many

39

into the array. Then the average energy of all the vectors
(500 vectors = 10 seconds of speech) is calculated for
future use as a threshold. Several alternative energy
thresholds were considered before the average energy was
considered an optimum threshold.

After energy normalizing each vector (by dividing each
vector element by the entire vector's energy), those
vectors with energy below the threshold energy are deleted.
This is accomplished in the subroutine LOWENERGY which also
assigns a weight of one to each vector prior to deletion.
Deleted vectors are then weighted zero. (This weighting
system dramatically reduces the number of calculations
previously used by Hussain to delete vectors). Vectors
with weights of zero are bubble sorted to the end of the
data array and the number of sixteen-dimensional vectors
left is calculated and stored in position IDATA(IDATA2+1)
(See CMPRS1.FR, Appendix C). If this number is less than
seventy the program returns to TRAIN. If not it calls the
subroutine COMPARE, which is a comparison and averaging
routine.

program can digitize. This is especially important to the various phases of the speech recognition system. A SWAP calls each program for digitizing ten seconds of speech for phoneme template creation (ATOD10), two seconds of speech for word phoneme represenations (ATOD2), and five seconds of speech for speech recognition (ATOD5). Each program creates a file (OUT2,OUT5,OUT10, respectively) in which data conversion values or integer voltages are stored. The data are stored sequentially, such that every set of sixteen integer values represents a vector of speech.

An important requirement for the correct processing of speech is that the input speech be between +5 and -5 volts. This level should be checked on the oscilloscope.

Phoneme Template Creation

A prompt from TRAIN requires the user to input ten seconds of speech, from which the set of phonemes will be derived. When the user says a predetermined phrase into a microphone, ATOD10 produces the file OUT10 containing 500 sixteen dimensional vectors. A call is then made to the subroutine TEMPLATE, which directs the creation of the phoneme template. OUT10 is opened and read into a common buffer in which the sampled data points are stored in the array IDATA. Then the energy in each vector is calculated and stored in the array TENERGY. (Figure 4 is a printout of this array) Each vector's energy is entered sequentially

37

for use in the system described herein required a considerable amount of modifications. Many of Hussain's individual programs appear as subroutines in the current program, making it necessary to provide strict continuity among them, and to develop interface routines so they can become efficient contributors to a unified whole. Furthermore, Hussain's programs require extensive interaction between the user and the machine, often with the human performing tasks which could be better handled automatically (e.g. the creation of phoneme templates which involves manually removing undesirable phonemes). Hussain's results could not be reproduced by other experimenters because his choice of values for some constants used in the algorithms was often completely arbitrary (at least the method for determining them was not revealed). In the interest of a more scientific approach, an attempt has been made in this thesis to derive values for these constants (energy threshold, distance threshold, error constants) from a more logical basis.

A/D Conversion

After the analog speech has gone through the preprocessor, it is converted to a digital signal by one of three programs: ATOD2, ATOD5, and ATOD10. The numbers in the program titles indicate how many seconds of speech each

V. RESULTS

From the very start, this research effort was hampered
by time constraints. The thesis report which served as a
starting point for this particular investigation was
incomplete and contained vague documentation making it
necessary to do some preliminary analysis in order to define
what had already been accomplished. This coupled with the
sheer magnitude of the task of developing a complete,
operational speech recognition system where none had existed
previously, resulted in the expiration of available time
before a thorough quantitative evaluation of the system's
performance could be made. Such an evaluation would require
the compilation of large amounts of data and systematic
tuning of parameters in search of the optimal setting for
the control dials. The fine tuning, although important, is
not really essential to the validation of this system as
"operational", and so the discussion in this chapter will
focus on the performance of the different components and the
way they are integrated into the system.

The programs developed by Ajmal Hussain (5) provided
the foundation for most of this research effort. (The major
thrust of the present treatment is to take the various
individual parts of the puzzle provided by Hussain(5) and
others (10) (7) and combine them into a complete speech
recognition system.) However the adaptation of his programs

35

closeness, value until there are at most ten phonemes left in the string. [An aside: Low energy vectors are considered noise and are represented as zero. Two adjacent zeroes are compressed to one zero, and one zero by itself is ignored, as are all zeroes appearing at the begining of the string (5).] The array SOUND, containing these phonemes, is then entered into the appropriate location in the matrix VOCAB. After all of the vocabulary words have been processed, PRINTREP prints the words along with their phoneme representations.

Recognition

After completing the training phase, TRAIN enters the recognition mode. It asks the user to input five seconds of speech through the dynamic microphone. Then the program swaps to ATOD5 for A/D conversion and creation of file OUT5. Next, TRAIN passes the phoneme template (PHON), the distance matrix (DIS), the vocabulary words (WORD), and the phoneme word representations (VOCAB) to subroutine SPEECH. The conversion values are read into a buffer, where the vectors are normalized (NORMALIZE) and the phonemes are extracted (EXTRACT). The CMPRS subroutine compresses adjacent phonemes, and RECOG calls Hussain's speech recognition algorithm. The recognized word strings which result are printed on both the screen and the line printer.

equation

$$location = m(m - 1)/2 + n. \qquad (8)$$

A call to PRINTDIS provides a printout of the distances between phonemes, normalized to 100.

After these two steps are completed, TRAIN determines phoneme representations for the words in the vocabulary. Vocabulary words are flashed onto the video screen, one at a time. A swap to ATOD2 enables the A/D conversion of two seconds of speech. Conversion values are subsequently stored in the file, OUT2.

Next, REP is called to read the eight blocks (2048 conversion values) into a buffer. As in TEMP, every 16 values represents a vector of sampled speech. After normalization (NORMALIZE), phonemes are extracted (EXTRACT) from the speech file. This means that the speech file is represented as a string of 250 phonemes. CMPRS goes through the phoneme string comparing each pair of adjacent phonemes. If the two are identical, they are compressed into one, if they are dissimilar the distance between them is examined, and in the case of a distance within the threshold, the phoneme having the higher number is deleted from the string. If the distance lies outside the threshold, no action is taken and the next pair of phonemes is examined. CMPRS continues reducing the phoneme string by increasing the

33

calculation.

Creating a phoneme template requires digitized speech input and numerous calculations and comparisons. TRAIN requests that the user provide 10 seconds of input speech. A swap with TRAIN calls ATOD10, which performs the A/D conversion and stores the data conversion values into a file named OUT10. Then TEMPLATE reads 16 blocks (8192 conversion values) from the file into a buffer. Every sixteen conversion values represents a sixteen-dimensional vector of speech. The vectors are normalized (NORMALIZE), checked for low energy (LOW ENERGY), and compressed (CMPRS1) by removing the low energy vectors. A comparison and averaging routine called COMPARE finds each vector's nearest neighbor vector. Subsequently, the two nearest neighbors are averaged and, following compression (CMPRS1), a new set of vectors is defined. This process repeats until a maximum of sixty-nine vectors is left. (Seelandt (10) showed 70 phonemes was a viable number for characterizing sounds in speech). These remaining vectors are stored in the array PHON for future use. The value PHON(1121) is the total number of vector phonemes.

Next, TRAIN calls DISTANCE, which creates a lower triangular matrix of the distances between phonemes. The matrix is stored in array DIS, in which the location between two phonemes m and n (where $m>n$) can be found by the

32

IV. Software Design

The computer algorithms designed for use in this thesis were designed modularly on a hierarchical basis. The motivation for this was the requirement for ease in system modification. The various components can be changed to accommodate various methods of phoneme template creation, word representation, and continuous speech recognition. Therefore, this thesis provides a valuable tool for future research in this area.

The primary driving program of the system is TRAIN. Having two phases (training and recognition), . it ties all of the components of a speaker-dependent recognition system together. This chapter provides a discussion of these components.

[NOTE: In addition to the main speech recognition algorithms, several support algorithms were written for A/D conversion, file manipulation, and creation of a vocabulary of words. The source codes for these algorithms are in Appendix C.]

Training

Training is accomplished through subroutine calls from the program TRAIN. In other words, TRAIN has a training mode in which a phoneme template, a distance matrix, and a word phoneme template are made. Each mode is a separate

31

word recognition is then performed as described above. Finally, error statistics are calculated for the entire string. For continuous recognition the entire process is repeated by choosing the second best guess for the first word, then the third best guess and so on. Folowing the final iteration, the string having the least total error is accepted as the correctly recognized sequence of words. The entire process repeats for the next phrase indicated by a set of adjacent zeroes.

Time Alignment

Because phonemes are compared on a one-to-one basis after compression, word duration is ignored and time alignment is unnecessary.

```
   34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
34  0
35 18  0
36 52 46  0
37 51 52 24  0
38 34 27 33 46  0
39 43 40 39 51 29  0
40 48 39 15 36 24 30  0
41 51 50 37 55 40 26 38  0
42 48 51 52 68 45 18 47 26  0
43 35 36 37 50 28 16 32 30 19  0
44 45 37 20 29 23 28 16 41 45 27  0
45 38 36 19 16 33 47 26 52 64 46 21  0
46 44 42 18 11 37 49 27 54 66 49 24 11  0
47 29 22 28 31 26 42 26 49 59 42 23 17 22  0
48 32 24 36 40 25 44 29 50 60 42 25 25 33 13  0
49 39 27 48 54 21 45 39 53 58 44 31 39 47 28 17  0
50 49 40 27 42 24 20 18 33 37 29 21 35 37 32 34 32  0
51 59 63 63 80 56 32 59 35 22 35 58 75 78 70 71 68 42  0
52 64 69 70 86 63 37 66 40 22 38 64 82 84 78 78 76 52 23  0
53 73 76 77 91 71 51 70 48 35 49 70 88 90 84 84 80 70 51 29  0
54 47 48 42 58 38 19 39 18 22 18 37 54 56 50 52 54 27 28 31 56  0
55 43 48 28 15 47 54 39 57 71 54 30 14 13 26 36 31 45 83 89 93 61  0
56 43 50 31 22 49 57 42 59 73 56 34 19 18 28 38 33 48 85 91 95 63  9  0
57 47 52 28 15 49 54 39 56 71 54 32 17 13 31 41 36 44 82 88 93 60 10  8  0
58 52 43 25 39 32 21 22 33 37 31 23 36 38 39 37 42 16 42 51 70 26 43 46 42  0
59 52 42 19 35 30 26 22 28 43 32 23 29 31 31 32 35 16 50 58 72 31 36 39 36 14  0
60 51 40 36 45 29 37 24 39 54 39 26 32 37 29 30 30 22 61 70 79 45 42 45 46 30 19  0
61 53 56 45 61 48 36 47 20 38 37 49 57 59 55 56 60 44 47 51 50 32 60 62 60 45 40 46  0
62 45 48 32 42 41 32 33 29 47 31 32 39 41 39 44 53 38 58 63 61 37 43 44 43 36 34 45 22  0
63 44 47 21 15 41 49 31 53 67 49 26 13 13 28 38 52 40 78 84 90 56 14 22 15 39 33 43 59 41  0
64 28 24 29 33 17 32 26 41 49 31 24 21 26 16 21 32 26 60 67 74 40 32 35 34 31 30 30 48 33 26  0
65 27 33 41 48 27 29 38 37 31 24 34 42 46 38 39 44 36 40 45 60 30 49 52 50 41 40 44 40 32 43 23  0
66 36 40 42 57 33 23 38 28 25 17 34 52 55 47 47 48 31 29 34 57 17 60 62 59 35 36 45 39 38 54 35 19  0
```

Figure 6b.   Distance Matrix (continued)

Word Phoneme Representation

Once the phoneme template and the distance matrix are formed, the system can start making word phoneme representations. TRAIN reads from a file the list of vocabulary words and stores them as Hollerith strings in the array named (appropriately) WORDS. The vocabulary used in this thesis is that shown in Appendix A. TRAIN's next step is to loop through the vocabulary words to find the phoneme representation of each.

The user is prompted to enter two seconds of speech by saying the word which appears on the computer video screen during the allotted time window. A swap to ATOD2 produces the file OUT2 containing the 125 digitized speech vectors.

TRAIN then calls the subroutine REP, which controls the procedure for finding that word's phoneme representation. As was the case in the creation of phoneme templates, the data vectors are normalized and the low energy vectors are deleted using the previously determined threshold. Each data vector is then compared to the complete set of phoneme vectors by calculating the distance between the data vector and each of the phoneme vectors. After this the phoneme vector whose distance calculation produced the smallest value is used to represent the string vector. This is repeated for the entire string of speech vectors, forming an array of numbers representing the closest phoneme vectors. This string is then compresesed (CMPRS) by replacing

43

identical phonemes with a single phoneme and deleting the highest-numbered adjacent phonemes whose distances are below the threshold distance, which is 10. A check is made to see if the resulting string has a maximum of ten phonemes. If not, the distance threshold is increased by 0.5 and the string is compressed again. (Note: the choice of 10 as a distance threshold is arbitrary as is the choice of a maximum of ten phonemes for word representation. A maximizing routine could probably be incorporated here to more accurately determine the values for these limits). Figure 7 shows the string of numbers representing the input speech vectors, before and after compression.

```
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0 23 34 34 66 66 38  0  0  0  0  0  0  0  0 52 52 52 61 61 61 61
26 26 26 26 26 25 25 27 28 31 37  0  0  0  0  0  0 20 20 20 14  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0


23 34 66 38 52 61 25 37 14  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 7. Phoneme String Before and After Compression

After the phoneme representations for all the vocabulary words have been found, they are placed in the array VOCAB. TRAIN then writes them to the file VOCABUL, and reads them from the same. TRAIN also prints the vocabulary words, along with their phoneme representations on the line printer. Figure 8 shows an example of this printout.

47

Word										
ADVISE	23	34	66	38	52	61	25	37	14	0
AFFIRMATIVE	58	6	8	52	43	28	31	0	0	0
AFT	16	65	25	65	25	0	0	0	0	0
AIR-TO-AIR	2	35	33	22	14	16	47	22	45	0
AIR-TO-SURFACE	23	33	5	14	22	14	0	0	0	0
ALPHA	37	62	61	41	54	12	36	0	36	0
ARM	63	41	61	62	43	25	27	0	13	0
BACKSPACE	11	61	16	20	37	14	29	14	13	0
BEARING	47	34	7	6	22	16	0	0	0	0
BRAVO	40	9	10	52	41	36	0	31	0	0
CANCEL	17	16	27	0	29	0	37	0	0	0
CHAFF	13	14	15	47	33	65	25	27	13	0
CHANGE	14	15	14	45	47	34	16	14	13	0
CHANNEL	14	15	0	16	22	16	22	0	0	0
CHARLIE	15	57	7	26	7	43	44	46	45	0
CLEAR	46	64	45	47	48	5	45	0	0	0
CONFIRM	21	7	6	44	27	0	0	0	0	0
DEGREES	37	21	22	2	56	57	20	0	0	0
DELTA	30	38	43	39	12	58	36	63	28	0
EAST	13	0	37	0	20	29	0	15	0	0
ECHO	16	28	0	45	44	45	31	0	13	0
EIGHT	1	17	46	0	0	0	0	0	0	0
ENTER	57	17	16	0	30	45	0	0	0	0
FAULT	12	41	54	39	12	0	37	0	0	0
FIVE	13	0	61	62	26	25	27	36	0	0
FLARES	44	38	48	5	3	45	15	14	19	0
FORWARD	59	12	40	37	0	37	0	13	0	0
FOUR	59	36	58	31	60	31	0	0	0	0
FOXTROT	41	26	62	27	29	14	57	13	0	0
FREQUENCY	30	48	2	17	37	0	0	0	0	0
FUEL	1	23	55	46	31	60	31	13	0	0
GUN	28	44	25	43	25	27	0	0	0	0
HEADINGS	27	64	34	38	28	16	47	2	20	0
HUNDRED	25	28	0	30	0	0	0	0	0	0
KNOTS	27	54	52	26	61	0	20	29	20	0
LOCK-ON	39	42	53	61	12	44	27	25	27	0
MAP	0	0	0	0	0	0	0	0	0	0
MARK	12	42	53	42	62	6	36	21	13	0
MILES	25	66	12	20	29	20	13	0	0	0
MINUS	62	43	26	65	16	63	20	13	0	0
MISSEL	63	22	45	46	63	20	29	31	31	0
NEGATIVE	22	34	35	47	0	0	0	0	0	0
NINE	27	66	43	26	25	22	16	46	0	0
NORTH	31	36	12	39	12	44	31	0	13	0
NOSE	28	38	7	8	32	37	14	29	20	0
ONE	31	50	12	42	39	25	27	36	0	0
POINT	13	36	12	11	12	25	27	20	0	0
PROFILE	44	8	10	40	31	12	41	29	37	0
RADAR	21	22	34	35	34	17	28	12	36	0
RANGE	21	33	34	17	18	24	0	14	15	0
REPORT	59	36	40	32	31	21	0	0	0	0
RHAW	31	12	54	41	25	58	0	0	0	0
SEARCH	20	0	46	44	6	30	0	14	15	0
SELECT	20	0	31	0	31	25	27	0	0	0
SEVEN	20	20	0	45	64	25	28	0	0	0
SIX	20	63	22	33	34	47	20	29	20	0
SMS	27	31	37	20	29	20	1	20	37	0
SOUTH	13	0	37	0	28	25	62	0	13	0
STATION	37	20	57	2	14	15	63	13	0	0
STRAFE	20	29	0	21	22	33	34	47	2	0
TAIL	45	35	64	47	2	19	28	36	0	0
TARGET	36	12	43	54	43	7	6	0	2	0
THOUSAND	25	54	26	62	61	41	12	31	20	0
THREAT	21	22	33	66	65	64	28	0	13	0
THREE	21	6	5	33	48	2	0	37	0	0
TWO	13	0	29	14	0	1	2	46	0	0
WAYPOINT	31	38	34	35	34	17	36	14	13	0
WEAPON	7	8	66	40	0	45	0	0	0	0
WEST	40	8	65	66	64	20	20	29	14	0
ZERO	20	1	24	6	5	6	40	31	0	0

Figure 8.   Word Phoneme Representations
from TRAIN

48

Speech Recognition

TRAIN enters the recoginition mode after the training session is completed, and prompts the user for a five second input of speech. Once again a SWAP call is used for A/D conversion and a file (OUT5) is written to contain sampled data points. The governing subroutine SPEECH then calls the necessary routines to read the data into the common buffer, normalize the data vectors, extract and compress the phoneme number string, and finally, perform speech recognition. All the steps prior to recognition are the same as those in word phoneme representation, with the exception of compression. Here, compression occurs only once, as there is no minimum number of phonemes required. The system is now ready to enter the recognition mode.

The method for recognition has alredy been discussed (see Word Recognition). Figure 9 shows a string of phoneme numbers representing five seconds of input speech. The second string represents the same speech vectors after compression has been performed. The zeroes represent points of low energy. If a zero appears alone, it is ignored in the recognition phase. If two are side-by-side, the recognition routine recognizes it as a word boundary, and starts counting the number of subsequent phonemes. If less than two phonemes are counted, the routine decides that it has made a mistake and starts counting again at the next boundary. If it counts more than nine adjacent phonemes, it

assumes that more than one word is present, so it goes into

the continuous recognition mode.

```
0  0  0  0  0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0 49  8  9 52 61 61 61 61 61 61 52  0
0  0  0 40 10 10 10 10  9 40 40 40 31  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0 14  0  0 27 43 65 43 43  9 10 43  7 28  0  0  0 45 45  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0


13  0 49  8  9 52 61 52  0 40 10  9 40 31  0 14  0 27 43 65 43  9 10 43  7
28  0 45  0  1  2 35  5  3  0 45 64 47 45  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 9.   Phoneme String of Five Seconds of Speech,
            Before and After Compression

To perform word recognition, phonemes are compared to

string numbers and the best fitting vocabulary words

determined, based on the distances between phonemes. The

algorithm's best guesses are printed on the video screen and

line printer attached to the Eclipse.

Figure 10 shows a sample phoneme string before and

after compression, along with the algorithm's best guess.

In this example, the word spoken was "ONE" and was spoken by

the system's trainer (a female). The computer recognized

the word correctly as "ONE", but then apparently went on to

identify ambient noise as "EIGHT".

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0 31 50 12 12 54 39 42 39 25 27 36  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
31 50 12 42 39 25 27 36  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
NINE      27  46  43  26  25  22  16  46   0   0
NORTH     31  36  12  39  12  44  31   0  13   0
NOSE      28  38   7   8  32  37  14  29  20   0
ONE       31  50  12  42  39  25  27  36   0   0
POINT     13  36  12  11  12  25  27  20   0   0
PROFILE   44   8  10  40  31  12  41  25  37   0
```

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0 36 10 51 51 53 53 53 43 45 27  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
36 10 51 53 43 45 27  0 13  0 13  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
ONE46
EIGHT
```

Figure 10.   "ONE" Spoken by KRD

Figure 11 shows a similar event; however, in this case
the word was "AIR-TO-AIR," spoken by a male speaker. The
minor inaccuracy of the machine's recognition ("CLEAR
CLEAR") in this case can be attributed to the fact that the
system had not been retrained to the male speaker's voice.

Another male speaker trained the system and entered the
word "AIR-TO-AIR". Both his phoneme string before and after
compression are shown in Figure 12, along with the computer
algorithm's best guess.

Similar tests were performed for continuous speech
recognition. The female speaker who originally trained the
system spoke the phrase, "ONE TWO THREE." Then, using her
training templates, the computer algorithm recognized "AFT
AFT ONE AFT FUEL AFT THREAT AFT AFT". Analyzing this string
of words, one could expect the word "AFT" to represent
ambient noise. The confusability of "FUEL" for "TWO" and
"THREAT" for "THREE" means that not all the sounds present
in each word are identified or matched. Those sounds with
higher energy are detected, while softer sounds or lower
energy phonemes may be deleted as a result of the energy
threshold.

```
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2 47
47 35 34 34 33 22  0  0  0  0 14  0  0  0  0  0  0  0 24 19 22 16 22 47 47 22
22 45 45 45 46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 2 39 33 22 14 16 47 22 45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

ADVISE	23	34	66	38	52	61	25	37	14	0
AFFIRMATIVE	58	6	8	52	43	28	31	0	0	0
AFT	16	65	29	65	25	0	0	0	0	0
AIR-TO-AIR	2	39	33	22	14	16	47	22	45	0
AIR-TO-SURFACE	23	33	5	14	22	14	0	0	0	0
ALPHA	37	62	61	41	54	12	36	0	36	0
ARM	63	41	61	62	43	25	27	0	13	0
CHARLIE	15	57	7	26	7	43	44	46	45	0
CLEAR	46	64	45	47	48	5	45	0	0	0
CONFIRM	21	7	6	44	27	0	0	0	0	0
DEGREES	37	21	22	2	56	57	20	0	0	0

```
 0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0 17 33 34 34 33 33  6  6  0 46  0  0  0  0  0  0
 0 57  0  0  0  0  0  0 63 64 33 33  5  5  5  6  6  6  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13  0 17 33 34 33  6 46  0 57  0 63 64 33  5  6  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
CLEAR
CLEAR
```

Figure 11. "AIR-TO-AIR" Spoken By MGC

```
0  1  0  0  0  0  1  0  1  0  0 65  0  0  1  0  0  0  0  0 | 1 68  3  0
0  5 51 51 51 34 34 34 34 34 34 34 28 27  0  0  0  0  0  0 | 0  0  0  2  0
0  0  3  0 62  0  0  0  0  1  0  0  2  1  0  0  0  0  0  0 | 0  0  0  0  1
0  0  0  0  1  0  1  0 40  0  0  0  2  1  0  0  0  0  0  0 | 0  0  0  0  0
1 51 34 28 27  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 0  0  0  0  0
```

```
ADVISE          1   0   0   0   0   0 |  0   0   0   0
AFFIRMATIVE     1  47   6  36  51  23 | 24   1   0   0
AFT             1  51  34  28  27   1 |  0   0   0   0
AIR-TO-AIR      1  10  23   1   8   9 | 30  10   1   0
AIR-TO-SURFACE  1   8   9  36   3  17 | 21  17   1   0
ALPHA           1  32  29   1  32  27 |  1   0   0   0
ARM             1  28  51  47  36  21 |  1   0   0   0
CHANGE          1  24  18   6   1  18 |  1   0   0   0
CHANNEL         1  18  20   8  36   8 | 36  27   1   0
CHARLIE         1  17  36  12  32  15 | 32  29   1   0
CLEAR           1  36   9   8   9  27 |  1   0   0   0
CONFIRM         1  23  36  38  36  38 | 36  36   1   0
THOUSAND        1  19  37  46  33  15 | 30  31  17   0
THREAT          1  36  12  13  14  15 | 13  13   1   0
THREE           1  21  58   8   7  20 |  1   0   0   0
TWO             1  23  36  21   1   0 |  0   0   0   0
WAYPOINT        1  31   9   8   3  28 | 29  23   1   0
```

```
0  0 64  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  1 | 0  0  0  0  0
0  0  0  0  0  0  1  0  1  0  0  0 37  0  8  9  9 10 12 11 |11 36  0  0  0
0  0  0  0  0  9 10  9 23  9  9  9 10 10 10 10 10 36      | 0 21  0 37  0  0
0  5  0  0  0  0  1  1  0  0 99  0  0  0  0  0  0  0      | 0  0 64 40  2  0
0  0  0  0  0  0  0  0  0 37  0 99  0 99  2  0  0  0      | 0  0  3  0 64 64
64  0  5  0  1  0  1  0 37  8  9 10 12 11 36  0 10        | 9 10  9 23  9 10 36
21 37  0  5  0  1  0 99  0 37 99  2  0  3 64  0          | 1  0 64  0 92  1  0
1  0 92  1  0  1  0  1 40 42 67  1  2  0 99  0 68         | 1  0 63  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          | 0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          | 0  0  0  0  0  0
ADVISE
TWO66
CLEAR
```

Figure 12.   AIR-TO-AIR Spoken by DS

In summary, several trials were made using the system trained by both a male and a female speaker. The machine obviously has problems dealing with ambient noise, and usually associates its best guess phoneme string with "AFT" or "EIGHT". Also stops and pauses between syllables such as in "WAYPOINT" appear as zeroes. If the pause duration was too long the system determined that two words were spoken instead of one, and an attempt at best fit was made for two words. The other trend of confusing similar sounding words reveals that vowels are easily spotted, however, low energy fricatives are difficult to detect. This could possibly be remedied by experimenting with voltage thresholding, or the system might be made to be self adjusting. The limitation of phoneme representations to 10 phonemes may be too restrictive, and so can be introducing an excessive number of errors by taxing the systems's inability to properly detect words which are not well suited for such a compression.

VI. Conclusions

Although the system has not been thoroughly tested, several conclusions can still be drawn from the data already accumulated concerning this speech recognition system. In addition, some specific recommendations can be given for system improvement. This chapter summarizes these two areas.

Summary

The speech recognition system which was designed and implemented was speaker dependent and operated near real time (after training). Several techniques were incorporated to characterize phonemes as vectors in space. Through the use of distance rules it was possible to characterize words by a phoneme representation, which could subsequently be used in word recognition. This approach to speech recognition offers several possibilities for future investigation such as varying the Minkowski distances, and the application of clustering techniques.

The system which was designed was user friendly, providing a system of recognition which needed little user interaction. The instructions were kept to a minimum and made easy to understand, thereby taking a lot of the guess work out of trying to understand the programmer's jargon.

The modularity of the system also proved useful by allowing easy modification through the use of comment characters. Any stage of recognition can be changed by simply altering a subroutine. Another advantage to the way this system was designed is that the variables which are passed are kept to a minimum, thus reducing the number of items which must be accounted for.

The objective of designing a speech recognition system capable of operating in real time was met by this research effort. The resulting system uses a phoneme set unique to a particular speaker and partial template matching for continuous word recognition.

Recommendations

Several recommendations can be made for improving system performance. First, the method of energy thresholding should be investigated to determine an optimum range of threshold values, since the present method allows some of the low energy vowels and most consonants to be deleted. Second, word phoneme representations probably should not be limited to a length of ten. Instead, natural breaks should be retained. Third, other methods of word boundary detection could take the arbitrariness out of the present system. Specifically, the use of two adjacent zero energy vectors as an arbitrary word-word boundary could be
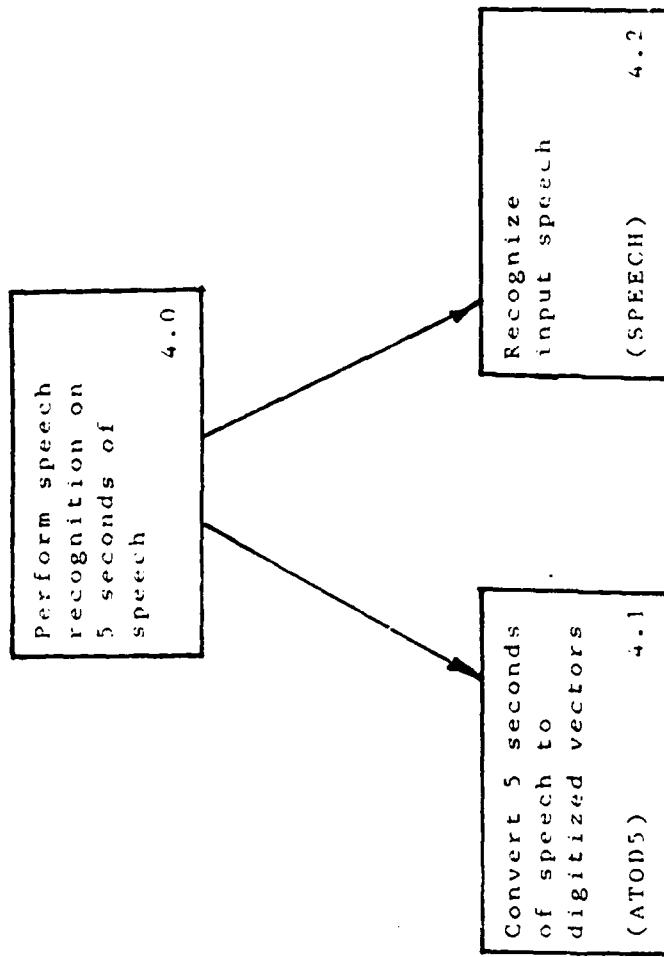
Figure 21. Structure Chart 4.0. Perform Speech Recognition
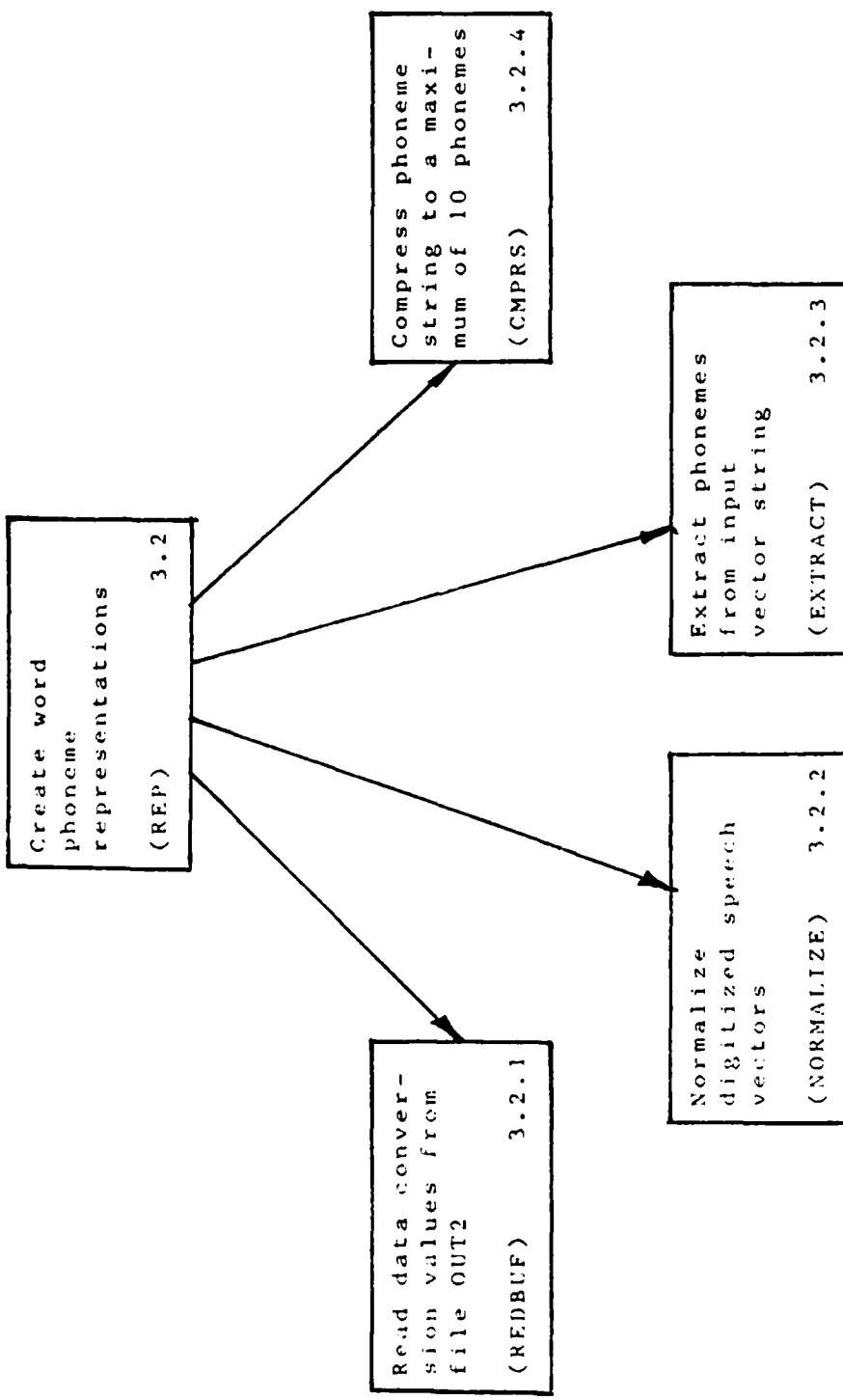
Figure 29. Structure Chart 3.2: Create Word Phoneme Representations
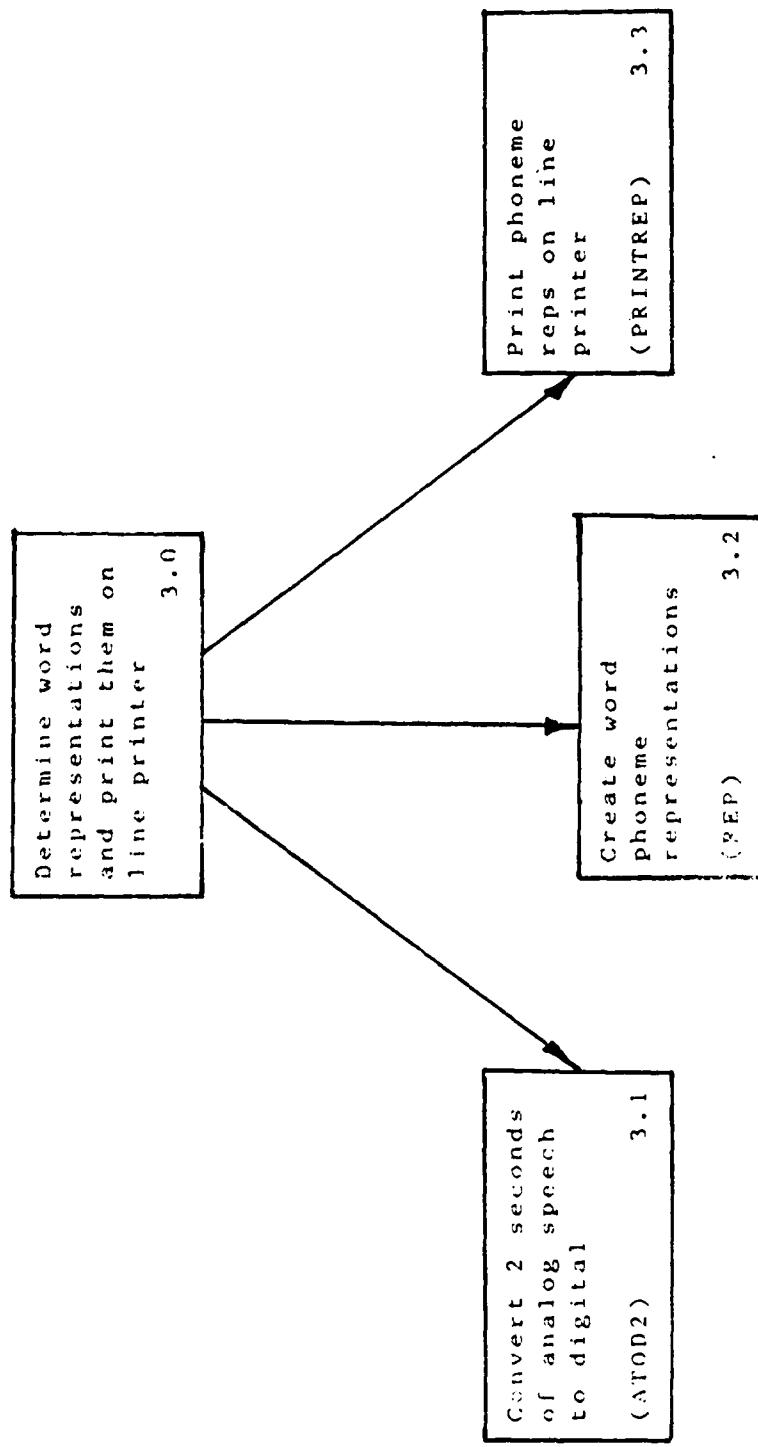
Figure 19. Structure Chart 3.0: Determine and Print Word Representations

69
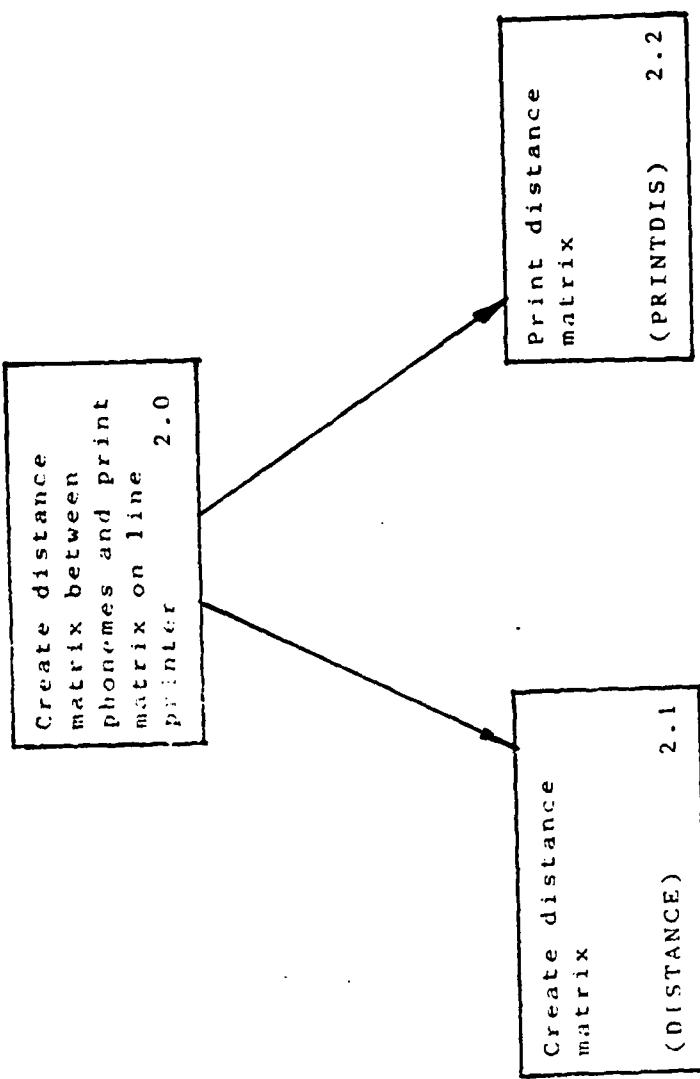
Figure 18. Structure Chart 2.0: Create and Print Distance Matrix
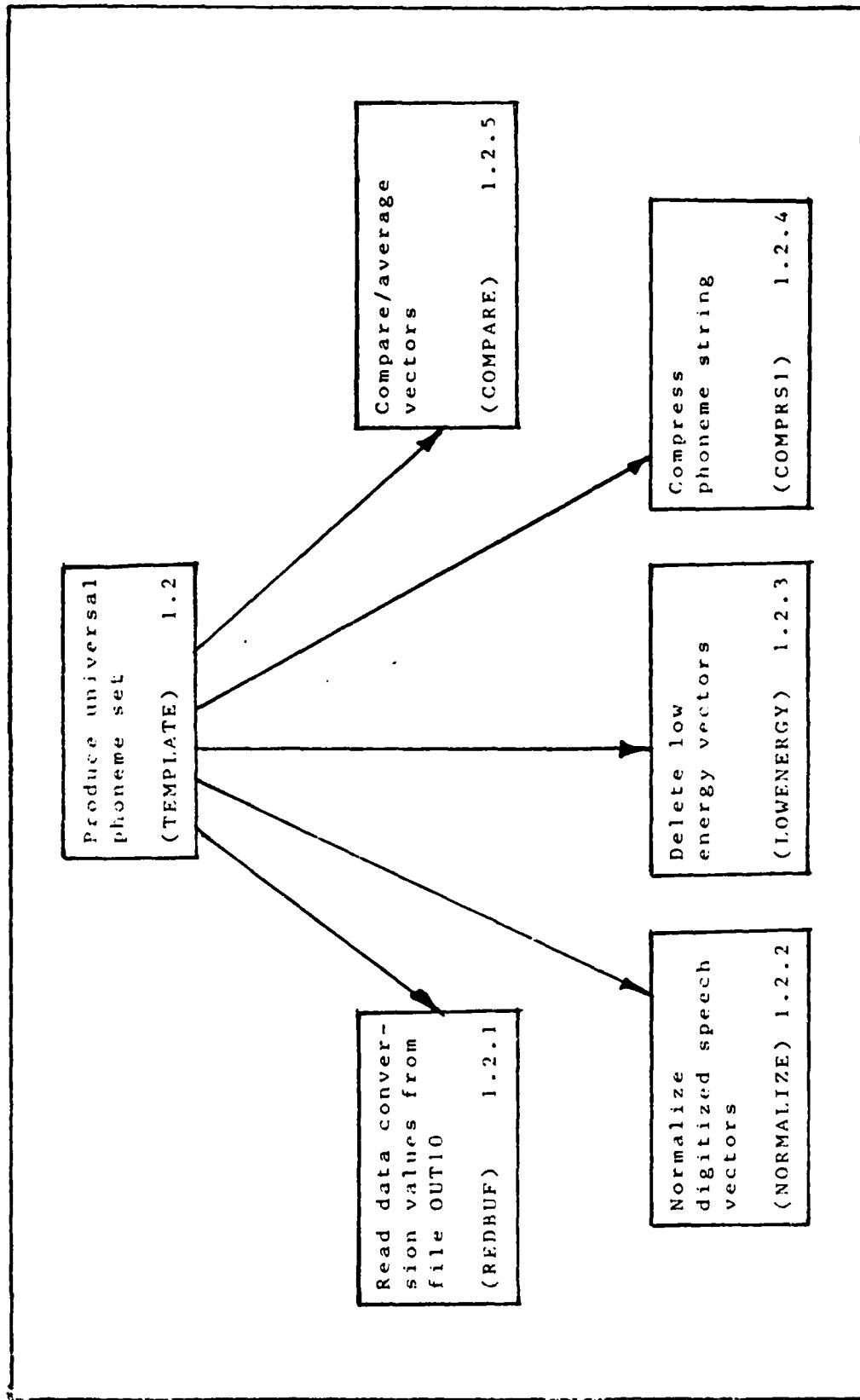
Figure 17. Structure Chart 1.2: Produce Universal Phoneme Set
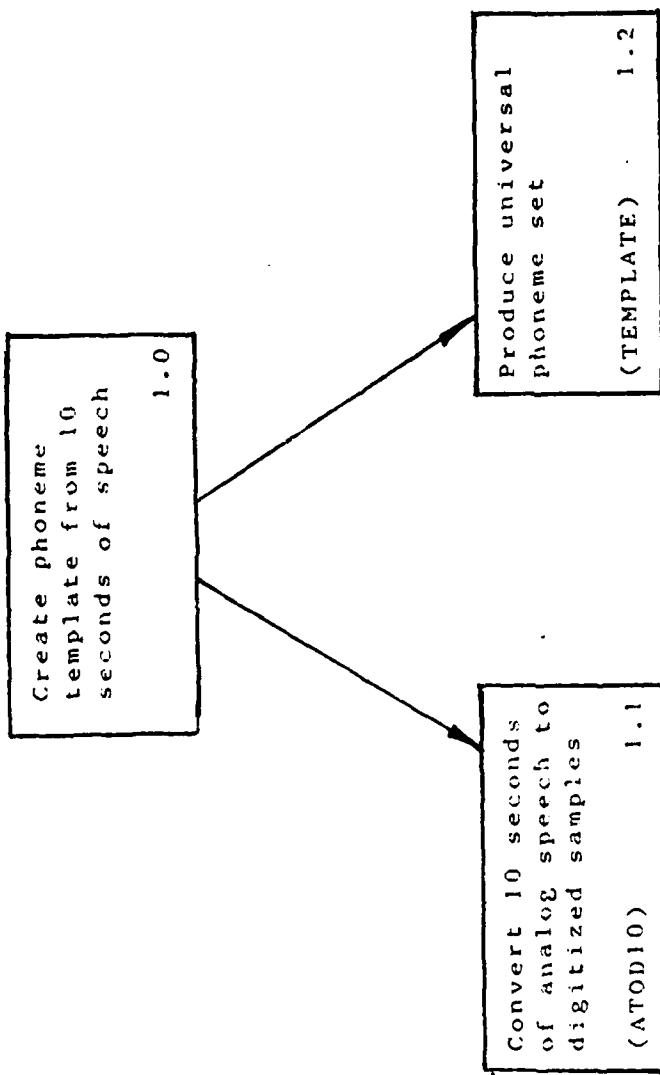
Figure 16. Structure Chart 1.0: Create Phoneme Template
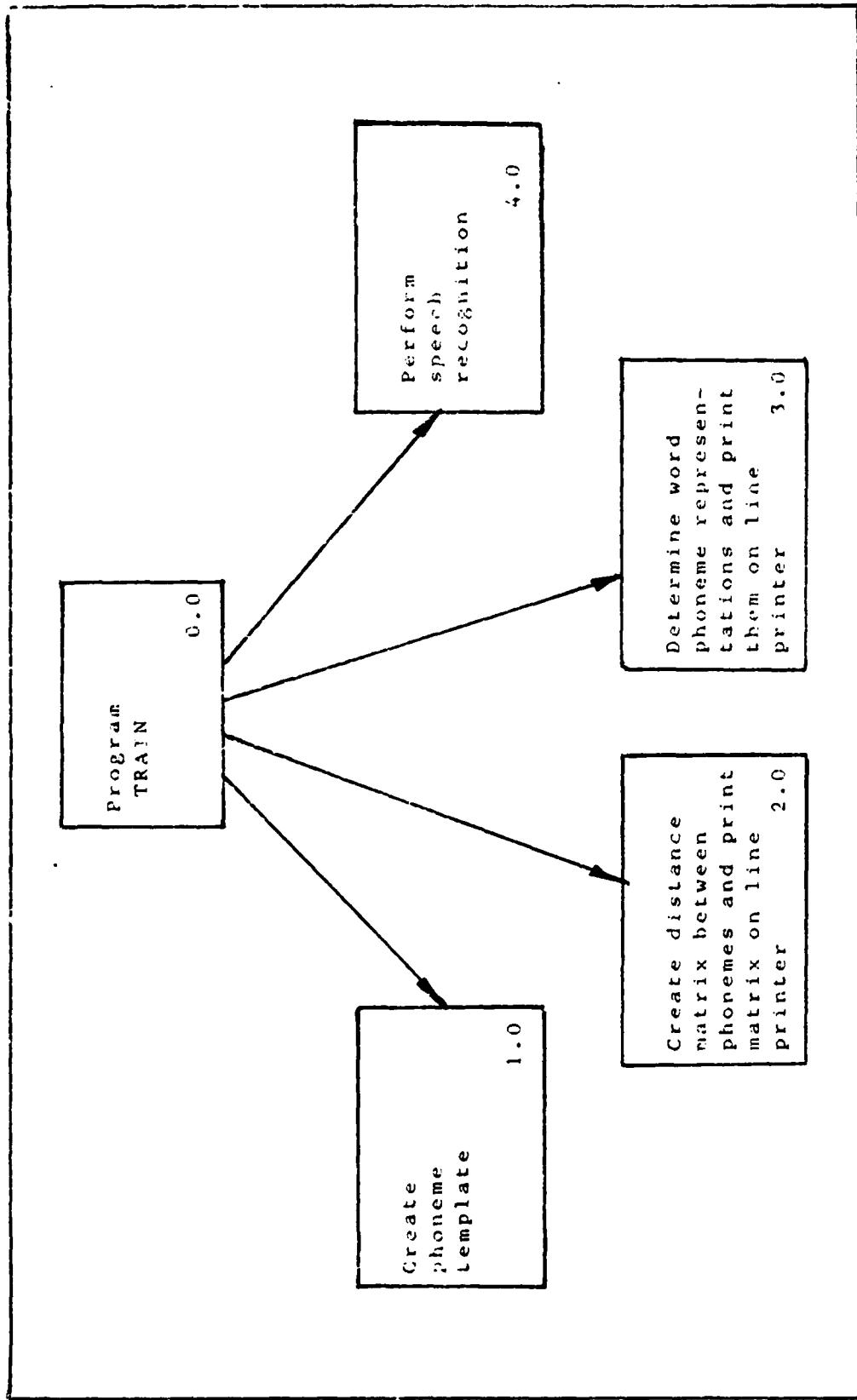
Figure 15. Structure Chart 0.0: Program TRAIN

Appendix B: Computer Program Structure Charts

Structure charts are included to provide a visual description of the proposed model for this speech recognition system. The modularity of the system allows simple modification at any location in the program.
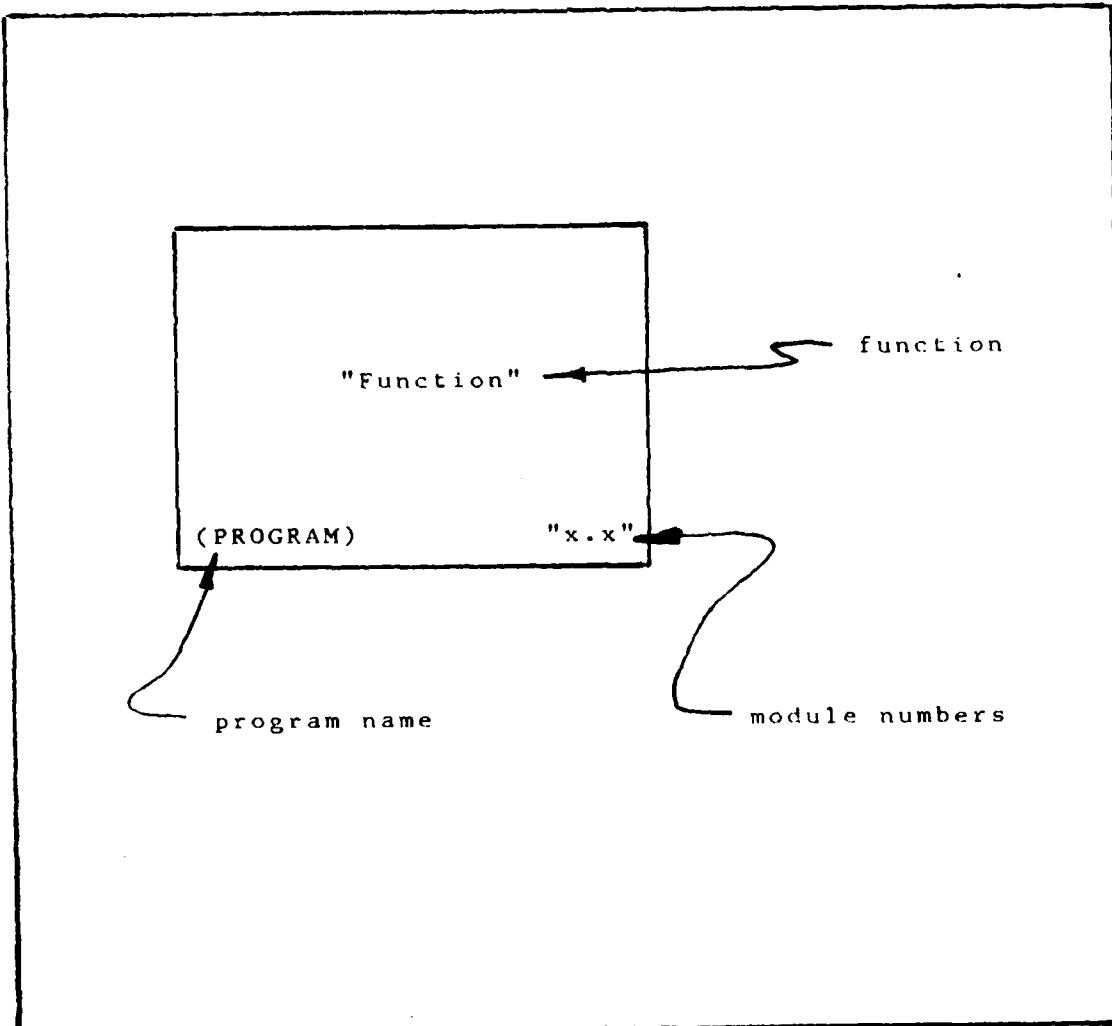
Figure 14. Legend for Structure Charts

APPENDIX B

COMPUTER PROGRAM STRUCTURE CHARTS

Though this speech recognition system can accommodate several different vocabularies, the seventy words below best represent the vocabulary necessary to a military environment.

ADVISE	DELTA	MAP	SEVEN
AFFIRMATIVE	EAST	MARK	SIX
AFT	ECHO	MILES	SMS
AIR-TO-AIR	EIGHT	MINUS	SOUTH
AIR-TO-SURFACE	ENTER	MISSILE	STATION
ALPHA	FAULT	NEGATIVE	STRAFE
ARM	FIVE	NINE	TAIL
BACKSPACE	FLARES	NORTH	TARGET
BEARING	FORWARD	NOSE	THOUSAND
BRAVO	FOUR	ONE	THREAT
CANCEL	FOXTROT	POINT	THREE
CHAFF	FREQUENCY	PROFILE	TWO
CHANGE	FUEL	RADAR	WAYPOINT
CHARLIE	GUN	RANGE	WEAPON
CHANNEL	HEADING	REPORT	WEST
CLEAR	HUNDRED	RHAW	ZERO
CONFIRM	KNOTS	SEARCH	
DEGREES	LOCK-ON	SELECT	

Figure 13.   AFTI-F16 Tentative Vocabulary (2)

APPENDIX A

AFTI-F16 VOCABULARY

Bibliography

1. Allen, Gordon R. _Expansion of the Eclipse Digital Signal Processing System_. MS Thesis GE/EE/82D-16. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1982 (A124 750).

2. Anderson, Timothy R., Electrical Engineer, personal interview by 1LT David B. Stockton. Air Force Aerospace Medical Research Laboratory, Wright-Patterson AFB OH, June 1984.

3. Doddington, George R. and Thomas B. Schalk. "Speech Recognition: Turning Theory to Practice," _IEEE Spectrum_, _18_: 26-32 (September 1981).

4. Haton, Jean-Paul. "Speech Recognition and Understanding," _Proceedings of the Sixth International Conference on Pattern Recognition_: 570-579. Institute for Electrical and Electronics Engineers, New York, June 1982.

5. Hussain, Ajmal. _Limited Continuous Speech Recognition by Phoneme Analysis_. MS Thesis GE/EE/83D-31. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1983 (A138 021).

6. Levinson, Stephen E. and Mark Y. Liberman. "Speech Recognition by Computer," _Scientific American_, _244_: 64-76 (April 1981).

7. Montgomery, Gerard J. _Isolated Word Recognition Using Fuzzy Theory_. MS Thesis GE/EE/82D-74. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1982 (A124 851).

8. Potter, R. K., George Kopp, and Harriet Green. _Visible Speech_. New York: D. Van Nostrand Company, Inc., 1947.

9. Rabiner, Lawrence R., Stephen E. Levinson, Aaron E. Rosenberg, and Jay G. Wilpon. "Speaker-Independent Recognition of Isolated Words Using Clustering Techniques," _IEEE Transactions on Acoustics, Speech, and Signal Processing_, _ASSP-27_: 336-349 (August 1979).

10. Seelandt, Karl G. _Computer Analysis and Recognition of Phoneme Sounds in Connected Speech_. MS Thesis GE/EE/81D-53. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1981.

This system was intended to present a model for a user friendly continuous speech recognition system. Though it is by no means a perfect system, it should provide a firm foundation for future studies.

changed to a more fuzzy determination. Finally, branching techniques could be used to predict phoneme transitions.

The continuation of this research should not be difficult. Investigations into energy thresholding should be made so that the system might become self-adjusting. Additionally, other methods (besides Hussain's choice of Minkowski 4) of distance calculations should be studied to which will yield the best results. Also, the compression technique for extracted phonemes could be investigated to determine an optimum threshold for adjacent phoneme distance and to see if deleting the highest-numbered vector is really necessary. Montgomery(8) applied techniques involving average branching factors for phoneme transitions which could easily be incorporated into this system. Though ambient noise was not removed in the manner originally described, the effect of background noise on energy thresholded vectors is another subject worthy of further investigation. The choice of vocabulary for this system may not have been the best suited for it. Perhaps the phonetic alphabet (ALPHA, BRAVO, CHARLIE,...) would have more validity as these characters are easily distinguishable, having been designed for low confusability in the presence of high levels of background noise. Finally, the entire system would probably work more quickly if implemented on an array processor, since most of the calculations incorporate vector combinatorics.
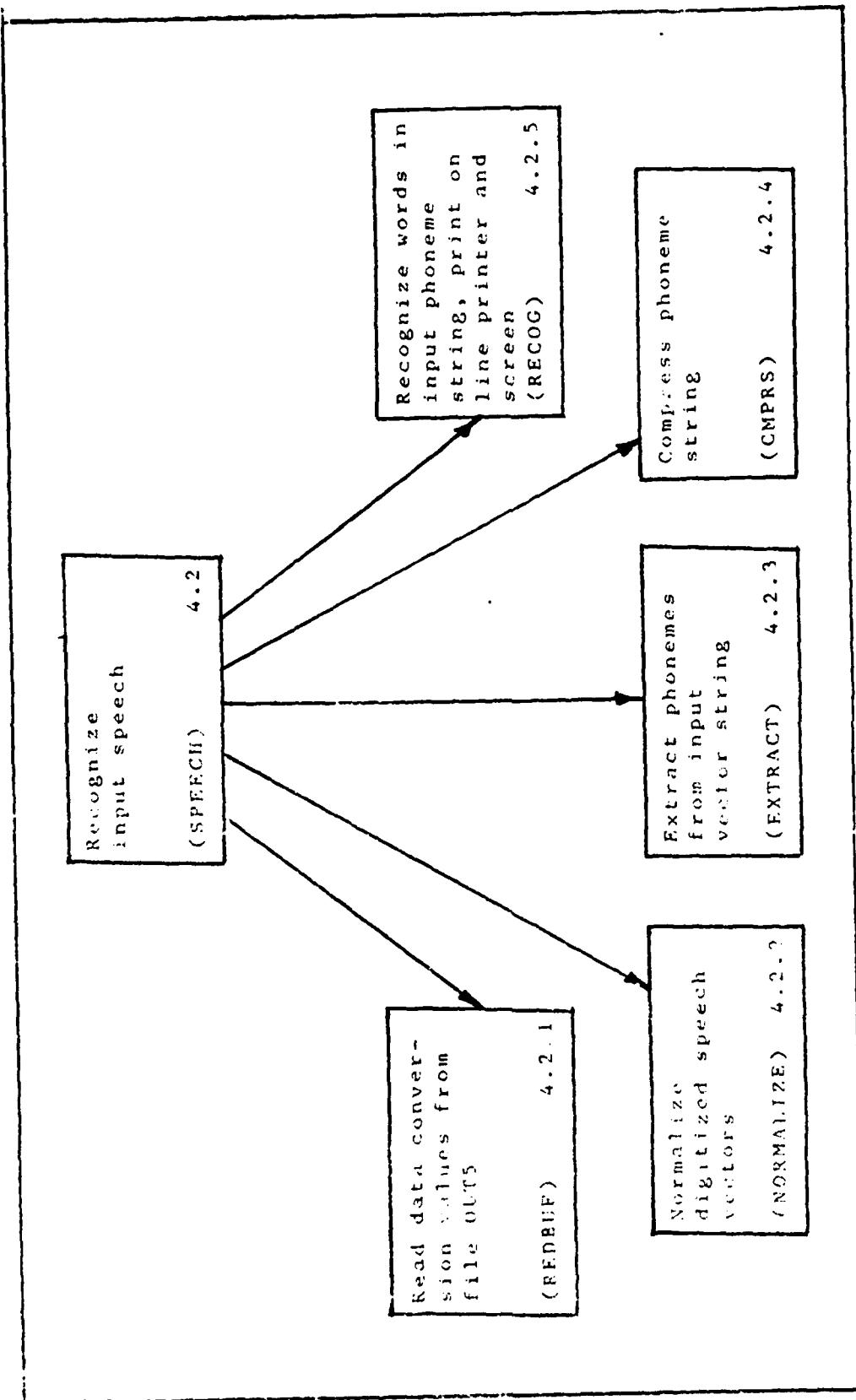
Figure 22. Structure Chart 4.2: Recognize Input Speech

APPENDIX C

COMPUTER PROGRAM LISTINGS

Appendix C: Computer Program Listings

The source code for all computer programs and support routines, written in FORTRAN V, is included in this Appendix. Two macrofiles ATODMC and TRAINMC are used to load all routines necessary for speech recognition. These programs appear in the following sequence:

```
*****ATODMC.MC*****************************************************

        Function: load ATOD programs required by TRAIN.

*****************************************************************


RLDR/P 2/K ATOD10 SAMCONFIG3 @SAMLIB@

RLDR/P 2/K ATOD5 SAMCONFIG3 @SAMLIB@

RLDR/P 2/K ATOD2 SAMCONFIG3 @SAMLIB@
```

```
C***********************************************************
C
C        Title:   ATOD2.FR
C        Author:  lLt Kathy R. Dixon
C        Date:    Nov 84
C
C        Function:  performs A/D on Eclipse for 5 sec
C
C        Command Line:
C
C              RLDR/P 2/K ATOD2 SAMCONFIG3 @SAMLIB@
C
C***********************************************************

        EXTERNAL IDS21                  ;external input device
        EXTERNAL IDS23                  ;external output device
        COMMON/IBUFF/IDATA3(16384)      ;input data buffer
        COMMON/IBUFO/IWAST              ;output data buffer

        INTEGER IORBA(16),DEVICE

        DEVICE=21                        ;input device
        IDATA1=61700K                    ;external clock
        IDATA2=1600                      ;conversion count

        TYPE"<CR>
    *   start<BEL><CR>"

        CALL DSTRT(IER)                 ;initialize A/D device
        IF(IER.NE.1)CALL ERROR("DSTRT ERROR")

        CALL DOITW(IORBA,IDS21,8,IDATA1,IDATA2,IDATA3,IER)
        IF(IER.NE.1)TYPE"DOIT ERROR",IER
        IF(IORBA(14).NE.40000K)TYPE"IORBA(14) RETURN",
    *   IORBA(14)


        TYPE"<CR>
    *   stop<BEL><CR>"

        CALL DFILW("OUT2",IER)
        IF((IER.NE.1).AND.(IER.NE.13))TYPE"DFILW ERROR",IER

        CALL CFILW("OUT2",2,IER)
        IF(IER.NE.1)TYPE"CFILW ERROR",IER

        CALL OPEN(1,"OUT2",2,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER

        CALL WRBLK(1,0,IDATA3,8,IER)
        IF(IER.NE.1)TYPE"WRBLK ERROR",IER
```

77

```
CALL CLOSE(1,IER)
IF(IER.NE.1)TYPE"CLOSE ERROR",IER

CALL EXIT
END
```

```
C************************************************************
C
C       Title:   ATOD5.FR
C       Author:  1Lt Kathy R. Dixon
C       Date:    Nov 84
C
C       Function:  performs A/D on Eclipse for 5 sec
C
C       Command Line:
C
C           RLDR/P 2/K ATOD5 SAMCONFIG3 @SAMLIB@
C
C************************************************************

        EXTERNAL IDS21                  ;external input device
        EXTERNAL IDS23                  ;external output device
        COMMON/IBUFF/IDATA3(16384)      ;input data buffer
        COMMON/IBUFO/IWAST              ;output data buffer

        INTEGER IORBA(16),DEVICE

        DEVICE=21                       ;input device
        IDATA1=61700K                   ;external clock
        IDATA2=4000                     ;conversion count

        TYPE"<CR>
     *  start<BEL><CR>"

        CALL DSTRT(IER)                 ;initialize A/D device
        IF(IER.NE.1)CALL ERROR("DSTRT ERROR")

        CALL DOITW(IORBA,IDS21,8,IDATA1,IDATA2,IDATA3,IER)
        IF(IER.NE.1)TYPE"DOIT ERROR",IER
        IF(IORBA(14).NE.40000K)TYPE"IORBA(14) RETURN",
     *  IORBA(14)


        TYPE"<CR>
     *  stop<BEL><CR>"

        CALL DFILW("OUT5",IER)
        IF((IER.NE.1).AND.(IER.NE.13))TYPE"DFILW ERROR",IER

        CALL CFILW("OUT5",2,IER)
        IF(IER.NE.1)TYPE"CFILW ERROR",IER

        CALL OPEN(1,"OUT5",2,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER

        CALL WRBLK(1,0,IDATA3,16,IER)
        IF(IER.NE.1)TYPE"WRBLK ERROR",IER
```

79

```
CALL CLOSE(1,IER)
IF(IER.NE.1)TYPE"CLOSE ERROR",IER

CALL EXIT
END
```

```
C*********************************************************************
C                                                                   *
C       Title:    ATOD10.FR                                         *
C       Author:   1Lt Kathy R. Dixon                                *
C       Date:     Nov 84                                            *
C                                                                   *
C       Function:  performs A/D on Eclipse for 10 sec.             *
C                                                                   *
C       Command Line:                                               *
C                                                                   *
C           RLDR/P 2/K ATOD10 SAMCONFIG3 @SAMLIB@                   *
C                                                                   *
C*********************************************************************
        EXTERNAL IDS21                      ;external input  device
        EXTERNAL IDS23                      ;external output device
        COMMON/IBUFF/IDATA3(16384)           ;input data buffer
        COMMON/IBUFO/IWAST                   ;output data buffer

        INTEGER IORBA(16),DEVICE

        DEVICE=21                                ;input device
        IDATA1=61700K       ·                    ;external clock
        IDATA2=8000                              ;conversion count

        TYPE"<CR>
    *   start<BEL><CR>"

        CALL DSTRT(IER)                  ;initialize A/D device
        IF(IER.NE.1)CALL ERROR("DSTRT ERROR")

        CALL DOITW(IORBA,IDS21,8,IDATA1,IDATA2,IDATA3,IER)
        IF(IER.NE.1)TYPE"DOIT ERROR",IER
        IF(IORBA(14).NE.40000K)TYPE"IORBA(14) RETURN",
    *   IORBA(14)

        TYPE"<CR>
    *   stop<BEL><CR>"

        CALL DFILW("OUT10",IER)
        IF(IER.NE.1.AND.IER.NE.13)TYPE"DFILW ERROR",IER

        CALL CFILW("OUT10",2,IER)
        IF(IER.NE.1)TYPE"CFILW ERROR",IER

        CALL OPEN(1,"OUT10",2,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER

        CALL WRBLK(1,0,IDATA3,32,IER)
        IF(IER.NE.1)TYPE"WRBLK ERROR",IER
```

```
CALL CLOSE(1,IER)
IF(IER.NE.1)TYPE"CLOSE ERROR",IER

CALL EXIT
END
```

```
C************************************************************
C
C        Title:   DISS.FR
C        Author:  1Lt Kathy Dixon
C        Date:    Nov 84
C
C        Function:   Prints distance matrix from a disk file.
C
C************************************************************


         REAL DIS(2432)

         CALL OPEN(1,"DIST",2,IER)
         IF(IER.NE.1)TYPE"OPEN ERROR",IER
         READ(1,100)(DIS(I),I=1,2432)
         WRITE(12,101)(DIS(I),I=1,2432)
100      FORMAT(G11.5)
101      FORMAT(8G11.5)
         CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE"CLOSE ERROR",IER

         RETURN
         END
```

```
C****************************************************************
C
C        Title:   PHONS.FR
C        Author:  1Lt Kathy Dixon
C        Date:    Nov 84
C
C        Function:  Reads up to 70 16-dimensional vectors
C                   from a file, PHONE and prints them on
C                   the line printer.
C
C****************************************************************


         INTEGER PHON(1130)

         CALL OPEN(1,"PHONE",2,IER)
         IF(IER.NE.1)TYPE"OPEN ERROR",IER
         READ(1,100)(PHON(I),I=1,1130)
         WRITE(12,101)(PHON(I),I=1,1130)
100      FORMAT(I6)
101      FORMAT(2X,16I6)
         CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE"CLOSE ERROR",IER

         RETURN
         END
```

```
C********************************************************************
C
C       Title:   REPS.FR
C       Author:  1Lt Kathy Dixon
C       Date:    Nov 84
C
C       Function:  Reads word phonemes from a file
C                  VOCABUL.
C
C********************************************************************


        INTEGER VOCAB(10,70),NOWORDS
        NOWORDS=70

        CALL OPEN(1,"VOCABUL",2,IER)
        IF(IER.NE.1)TYPE"OPEN ERROR",IER
        READ(1,100)((VOCAB(I,J),I=1,10),J=1,NOWORDS)
        WRITE(12,101)((VOCAB(I,J),I=1,10),J=1,NOWORDS)
100     FORMAT(I3)
101     FORMAT(10I3)
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE ERROR",IER

        RETURN
        END
```

END

FILMED

DTIC
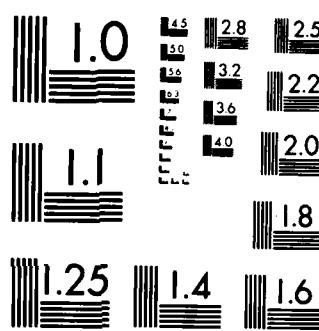
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

```
******TRAINMC.MC*******************************************

      Function: Loads TRAIN and required subroutines

************************************************************


RLDR TRAIN NEWSCR TEMPLATE DISTANCE PRINTDIS REDWRDS REP^
          PRINTREP SPEECH REDBUF NORMALIZE LOWENERGY CMPRS1^
          COMPARE EXTRACT CMPRS RECOG FINDWORD VTYPE^
          REDDIS REDPHON REDREP WRTDIS WRTREP WRTPHON @FLIB@
```

```
C****************************************************************
C
C          Title:  CMPRS.FR
C          Author: Capt. Ajmal Hussain
C                  Modified by 1Lt Kathy Dixon
C          Date:   Nov 84
C
C          Function:
C             Compresses input speech vectors after phoneme
C                  extraction.
C
C****************************************************************

           SUBROUTINE CMPRS(IDATA2,PHON,DIS,LIB,ISWITCH,SOUND,J)

           INTEGER SOUND(250),LIB(250),PHON(1130),IDATA2,ISWITCH
           REAL DIS(2432),LDIS


           NOS = (1+ISWITCH)*125                ; no. of vectors for
                                                ;processing
           LDIS=10.0                            ;set distance threshold

           DO 76 I=1,250
76         SOUND(I)=0

C          WRITE(12,35)(LIB(I),I=1,NOS)         ;option to write
35         FORMAT(25I3)              ;phoneme string to line printer

1          DO 806 K=1,5
           DO 809 I=1,((IDATA2/16)-1)
           IF(LIB(I).EQ.0)GO TO 807
           IF(LIB(I+1).EQ.0)GO TO 807
           IF(LIB(I).EQ.LIB(I+1))GO TO 807
           N=LIB(I)
           P=LIB(I+1)
           IF(N.GT.P)GO TO 810
           Q=N
           N=P
           P=Q
810        IF(DIS(((N*(N-1))/2)+P).GE.LDIS)GO TO 807
           IF(LIB(I).LT.LIB(I+1))LIB(I+1)=LIB(I)
           IF(LIB(I).GT.LIB(I+1))LIB(I)=LIB(I+1)
807        CONTINUE
809        CONTINUE
806        CONTINUE

           J=1
           DO 805 I=1,(IDATA2/16)
           IF((LIB(I).EQ.0).AND.(J.EQ.1))GO TO 805
           IF((LIB(I).EQ.0).AND.(LIB(I-1).NE.0))GO TO 805
```

```
           IF(LIB(I).EQ.LIB(I+1))GO TO 805
           LIB(J)=LIB(I)
           J=J+1
805        CONTINUE
           DO 808 L=J,(IDATA2/16)
808        LIB(L)=0

           IF(ISWITCH.EQ.1)GO TO 1001           ;check to see if
           LDIS=LDIS+0.5                         ;need to compress
           IF(J.LE.10)GO TO 1001                 ;to 10 phonemes
           GO TO 1              .                ;threshold

1001       DO 900 I=1,250                        ;store word phonemes
900        SOUND(I)=LIB(I)                       ;into SOUND

1000       CONTINUE
C          WRITE(12,35)(LIB(I),I=1,NOS)          ;option to print
                                        ;phonemes on line printer
                                        ;after compression

           RETURN
           END
```

```
C*********************************************************************
C
C         Title:  CMPRS1.FR
C         Author: 1LT Kathy Dixon
C                 Based on subroutine by CAPT Ajmal Hussain
C         Date:   Nov 84
C
C         Function:
C            Deletes vectors with weight 0.
C
C*********************************************************************

          SUBROUTINE CMPRS1(IDATA2,IWEIGHT,IWGHTT)

          COMMON/IBUFF/IDATA(8192)

          INTEGER IDATA2 ,IWEIGHT(500),IWGHTT(500)

          J=1
          M=1

          DO 1 I=1,500
1         IWGHTT(I)=0

          DO 2 I=1,IDATA2,16              ;check vector components
          IF(IWEIGHT(INT((I+16)/16)).EQ.0)GO TO 2

          DO 3 K=0,15
          IDATA(J)=IDATA(I+K)             ;replace deleting vector
3         J=J+1                                     ;with next vector

          IWEIGHT(M)=IWEIGHT(INT((I+16)/16))
          M=M+1

2         CONTINUE

          DO 4 I=M,500

          DO 5 K=1,16
5         IDATA(((I-1)*16)+K)=0

4         IWEIGHT(I)=0

          DO 6 I=1,500
6         IWGHTT(I)=IWEIGHT(I)

          RETURN
          END
```

```
C********************************************************************
C
C          Title:   COMPARE.FR
C          Author:  1Lt Kathy Dixon
C                   Nearest phoneme code derived from
C                   CAPT Ajmal Hussain
C          Date:    Nov 84
C
C          Function:
C             Compares normalized data vectors.  Finds each
C             vector's nearest vector.  Averages two vectors,
C             replaces lowest numbered vector with new vector,
C             sets vector components of second vector to 32000.
C
C********************************************************************

          SUBROUTINE COMPARE(IDATA2)

          COMMON/IBUFF/IDATA(8192)

          INTEGER IDATA5(500),NOVECT,IDATA2,TEMP4,IDATA6
          REAL DIFF(500)
          DOUBLE PRECISION REAL TEMP,TEMP1


          TEMP=0
          TEMP1=9.0E60

          IDATA6=IDATA(IDATA2+1)*16

          DO 104 J=1,IDATA6,16
          DO 102 K=1,IDATA6,16

          IF(J.EQ.K) GO TO 103

          DO 101 L=0,15
101       TEMP=TEMP+(FLOAT(IDATA(J+L))-FLOAT(IDATA(K+L)))**4


          IF(TEMP.GE.TEMP1) GO TO 103
          TEMP1=TEMP

          IDATA5(INT((J+15)/16))=INT((K+15)/16)
          DIFF(INT((J+15)/16))=TEMP
103       TEMP=0
102       CONTINUE

          TEMP1=9.0E60
104       CONTINUE
```

```fortran
      TEMP=0

      KKK=0

      NOVECT=IDATA(IDATA2+1)

      DO 107 I=1,NOVECT
107   IF(DIFF(I).GT.TEMP)TEMP=DIFF(I)

      DO 108 I=1,NOVECT
108   DIFF(I)=(DIFF(I)/TEMP)


      DO 111 I=1,NOVECT
      J=IDATA5(I)
      TEMP=DIFF(I)

      DO 2 JJ=1,NOVECT
      K=0
      IF(IDATA5(JJ).EQ.I)TEMP1=DIFF(JJ)
      IF((TEMP1.EQ.0).OR.(TEMP1.EQ.100))GO TO 2
      IF(TEMP1.LT.TEMP)K=JJ
      IF(J.EQ.K)GO TO 10
2     CONTINUE

      GO TO 111

10    DO 15 KL=1,16
      IDATA(((I-1)*16)+KL)=(IDATA(((I-1)*16)+KL)
     *  +IDATA(((J-1)*16)+KL))/2
15    IDATA(((J-1)*16)+KL)=32000

      DIFF(J)=100

      DO 20 JJJ=1,NOVECT
      IF((IDATA5(JJJ).EQ.I).AND.(DIFF(JJJ).NE.100))
     *  DIFF(JJJ)=0
20    CONTINUE

111   CONTINUE

      DO 222 I=1,NOVECT
222   IF(DIFF(I).EQ.100)KKK=KKK+1

      IDATA(IDATA2+1)=IDATA(IDATA2+1)-KKK


      RETURN
      END
```

91

```
C*******************************************************************
C
C         Title:  DISTANCE.FR
C         Author: Capt. Ajmal Hussain
C                 Modified by 1 Lt Kathy Dixon
C         Date:   Nov 84
C
C         Function:
C            Finds Minkowski 4 distance between phonemes in
C            phoneme template.
C
C*******************************************************************

          SUBROUTINE DISTANCE(PHON)

          REAL DIS(2432)
          INTEGER PHON(1130)
          DOUBLE PRECISION REAL TEMP, TEMP1

          DO 10 I=1,2432                    ;zero distance matrix
10        DIS(I)=0.

          TEMP=0
          TEMP1=0
          I=1
          DO 31 J=1,(PHON(1121)*16),16
          DO 32 K=1,(PHON(1121)*16),16
          IF(K.GT.J) GO TO 35
          DO 33 L=0,15
33        TEMP=TEMP+(FLOAT(PHON(J+L))-FLOAT(PHON(K+L)))**4
                                           ;M-4 calculation

          IF(TEMP.GT.TEMP1)TEMP1=TEMP    ;find largest distance
          DIS(I)=TEMP                            ;store distance
          I=I+1                                 ;increment DIS
35        TEMP=0
32        CONTINUE
31        CONTINUE
          DO 34 I=1,(((PHON(1121)*(PHON(1121)-1))/2)
     *    +PHON(1121))
34        DIS(I)=((DIS(I)/TEMP1)**0.25)*100  ;normalize DIS to
                                             ;TEMP1

          DIS(2416)=PHON(1121)              ;store no. phonemes

          CALL WRTDIS(DIS)

          RETURN
          END
```

```
C******************************************************************
C
C         Title:   EXTRACT.FR
C         Author:  Capt. Ajmal Hussain
C                  Modified by 1Lt Kathy Dixon
C         Date:    Nov 84
C
C         Function:
C            Extracts phonemes from input speech file.
C
C******************************************************************

          SUBROUTINE EXTRACT(IDATA2,PHON,IENERGY,ENRGY,LIB)

          COMMON/IBUFF/IDATA(4096)

          INTEGER LIB(250),PHON(1130)
          REAL IENERGY(500),ENRGY
          DOUBLE PRECISION REAL TEMP,TEMP1

          DO 1 I=1,250
1         LIB(I)=0                          ;zero array to hold
                                            ;extracted phonemes
          TEMP=0
          I=1
          M=0
          TEMP1=9.0E60

          DO 87 L=1,IDATA2,16

          IF(IENERGY(I).LT.ENRGY) GO TO 801     ;check energy
                                                ;threshold

          DO 86 K=1,(PHON(1121)*16),16

          DO 85 J=L,(L+15)
          TEMP=TEMP+(FLOAT(IDATA(J)-PHON(K+M)))**4    ;m-4
                                                      ;distance
85        M=M+1                                 ;between template

          IF(TEMP.GE.TEMP1)GO TO 82                   ;vector

          TEMP1=TEMP
          LIB(I)=(K+15)/16

82        TEMP=0
          M=0

86        CONTINUE
801       TEMP1=9.0E60
87        I=I+1
```

93

```
RETURN
END
```

94

```
C*******************************************************************
C
C       Title: FINDWORD.FR
C       Author: Capt. Ajmal Hussain
C               Modified by 1Lt Kathy Dixon
C       Date: Nov 84
C
C       Function:
C          This routine compares a phoneme string with word
C          strings in a library based upon a distance matrix
C          to give the word in the library which is the best
C          match.
C
C*******************************************************************

        SUBROUTINE FINDWORD(PHON,IPHON,VOCAB,MAT,TEMP3,L)

        INTEGER I,J,K,L,M,IPHON(250),LIB(700),NOWORD
        INTEGER VOCAB(10,70),PHON(1130)
        DOUBLE PRECISION REAL TEMP1,TEMP,TEMP3
        REAL MAT(2432),PEN

70      TEMP3 = 9.0E 60                         ;initialize variables
        TEMP = 0
        COUNT = 0
        NOWORD=70
        J=1
        L=0


        DO 1 I=1,70
        DO 2 K=1,10
        LIB(J)=VOCAB(K,I)
2       J=J+1
1       CONTINUE

C       start comparison

        DO 71 M = 1,(NOWORD*10),10      ;library, each word a
                                        ;maximum of 10 phonemes

        DO 72 K = -1,1          ;shift phoneme string one phoneme
                                ;left, none and one phoneme right to
                                ;account for error in first phoneme
                                                        ;string

        DO 73 I = 1,10          ;compare phoneme at a time for each
                                            ;word in library

        IF ((I+K).EQ.0) GO TO 73        ;skip first phoneme when
                                        ;string shifted left one
```

```
                                      ;phoneme

C        if both phonemes zero error value unchanged

         IF ((LIB(M+I-1).EQ.0).AND.(IPHON(I+K).EQ.0)) GO TO 73

C        if both phonemes not zero add distance between
C        phonemes to error value

         IF ((LIB(M+I-1).NE.0).AND.(IPHON(I+K).NE.0)) GO TO 74

C        if one phoneme zero only add penalty to error value

         P=LIB(M+I-1)
         PEN=0

         DO 135 JJ=0,15
135      PEN = PEN+ (FLOAT(PHON((P*16)-JJ))**4)
         PEN =((PEN)**0.25)

      TEMP1 = TEMP+PEN
      GO TO 75

  74     N = IPHON(I+K)                        ;find distance between
                                               ;phonemes from
         P = LIB(M+I-1)                        ;distance matrix
         IF(N.GE.P) GO TO 76
         Q = N
         N = P
         P = Q
  76     TEMP1 = MAT(((N*(N-1))/2)+P)

  75     TEMP = TEMP + TEMP1                    ;add distance to
         COUNT = COUNT + 1
  73     CONTINUE                               ;average error value
                                               ;and find word
         TEMP = TEMP/COUNT                     ;match with minimum
                                               ;error value

      IF(TEMP.GT.TEMP3) GO TO 77
      TEMP3 = TEMP
      L = M

  77     TEMP = 0                              ;initialize variables
                                               ;for  next word

      COUNT = 0
  72     CONTINUE
  71     CONTINUE

         RETURN
         END
```

```
C************************************************************
C
C         Title:  LOWENERGY.FR
C         Author: Capt. Ajmal Hussain
C                 Modified by Kathy Dixon
C         Date:   Nov 84
C
C         Function:
C            Checks energy normalized vectors for energy less
C            than a particular threshold. Sets vector weights
C            to 0.
C
C************************************************************

          SUBROUTINE LOWENERGY(IDATA2,IENERGY,ENRGY,IWG)

          INTEGER IDATA2,IWG(500)
          REAL IENERGY(500),ENRGY

          COMMON/IBUFF/IDATA(8192)

          K=0

          DO 1 I=1,500
1         IWG(I)=1

          DO 2 L=1,(IDATA2/16)
          IF(IENERGY(L).GE.ENRGY) GO TO 2          ;check phoneme
          IWG(L)=0                                      ;energy
          K=K+1
2         CONTINUE

          IDATA(IDATA2+1)=(IDATA2/16)-K            ;store no. of
                                                   ;REMAINING VECTORS

          RETURN
          END
```

```
C*********************************************************************
C
C         Title:   REP.FR
C         Author:  1Lt Kathy R. Dixon
C         Date:    Nov 84
C
C         Function:
C             Creates phoneme representation of a word.
C             Stores representation in array, SOUND.
C
C*********************************************************************

          SUBROUTINE REP(ENRGY,PHON,DIS,SOUND)

          COMMON/IBUFF/IDATA(4096)

          INTEGER PHON(1130),SOUND(250),LIB(250),IDATA2
          INTEGER ISWITCH,ISTOP,IFILE
          REAL DIS(2432),IENERGY(500),ENRGY

          IDATA2=1600                          ;conversion count
          ISTOP=8                              ;last block to read
          IFILE=2                              ;file to read
          ISWITCH=0                            ;switch to reduce input
                                               ;phoneme string to max 10
          CALL REDBUF(IFILE,ISTOP)             ;read data from file

          CALL NORMALIZE(IDATA2,IENERGY)       ;normalize vectors

          CALL EXTRACT(IDATA2,PHON,IENERGY,ENRGY,LIB) ;extract
                                                      ;phonemes

          CALL CMPRS(IDATA2,PHON,DIS,LIB,ISWITCH,SOUND,J)
                                               ;compress phoneme string

          RETURN
          END
```

```
C****************************************************************
C
C        Title:  REDWRDS.FR
C        Author: 1Lt Kathy R. Dixon
C        Date:   Nov 84
C
C        Function:
C           Reads vocabulary words from a file, WORDS.
C
C****************************************************************

         SUBROUTINE REDWRDS(NOWORDS,WORD)

         INTEGER WORD(7,70),NOWORDS

         CALL OPEN(1,"WORDS",2,IER)
         IF(IER.NE.1)TYPE"OPEN ERROR",IER
         READ(1,100)(WORD(1,I),I=1,NOWORDS)
100      FORMAT(S14)
         CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE"CLOSE ERROR",IER

         RETURN
         END
```

```
C***************************************************************
C
C          Title:  REDREP.FR
C          Author: Kathy Dixon
C          Date:   Nov 84
C
C          Function:  Reads word phonemes form a file.
C
C***************************************************************

          SUBROUTINE REDREP(NOWORDS,VOCAB)

          INTEGER VOCAB(10,70),NOWORDS

          CALL OPEN(1,"VOCABUL",2,IER)
          IF(IER.NE.1)TYPE"OPEN ERROR",IER
          READ(1,100)((VOCAB(I,J),I=1,10),J=1,NOWORDS)
100       FORMAT(I3)
          CALL CLOSE(1,IER)
          IF(IER.NE.1)TYPE"CLOSE ERROR",IER

          RETURN
          END
```

```
C****************************************************************
C
C        Title:  REDPHON.FR
C        Author: Kathy Dixon
C        Date:   Nov 84
C
C        Function:  Reads 70 16-dimensional vectors
C                          from a file PHONE.
C
C****************************************************************

         SUBROUTINE REDPHON(PHON)

         INTEGER PHON(1130)

         CALL OPEN(1,"PHONE",2,IER)
         IF(IER.NE.1)TYPE"OPEN ERROR",IER
         READ(1,100)(PHON(I),I=1,1130)
100      FORMAT(I6)
         CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE"CLOSE ERROR",IER

         RETURN
         END
```

```
C********************************************************************
C
C        Title:   REDDIS.FR
C        Author:  1Lt Kathy Dixon
C        Date:    Nov 84
C
C        Function:   Reads distance matrix from a disk file.
C
C********************************************************************

         SUBROUTINE REDDIS(DIS)

         REAL DIS(2432)

         CALL OPEN(1,"DIST",2,IER)
         IF(IER.NE.1)TYPE"OPEN ERROR",IER
         READ(1,100)(DIS(I),I=1,2432)
100      FORMAT(G11.5)
         CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE"CLOSE ERROR",IER

         RETURN
         END
```

```
C********************************************************************
C
C          Title:  REDBUF.FR
C          Author: 1Lt Kathy R. Dixon
C          Date:   Nov 84
C
C          Function:
C              Reads A/D data from a file.
C
C********************************************************************

           SUBROUTINE REDBUF(IFILE,ISTOP)

           INTEGER IFILE,ISTOP

           COMMON/IBUFF/IDATA(8192)

           IF(IFILE.EQ.10)GO TO 8                          ;read OUT10

           IF(IFILE.EQ.5)GO TO 7                           ;READ OUT 5

           CALL OPEN(1,"OUT2",2,IER)
           IF(IER.NE.1)TYPE"OPEN ERROR",IER

           GO TO 9

8          CALL OPEN(1,"OUT10",2,IER)
           IF(IER.NE.1)TYPE"OPEN ERROR",IER

           GO TO 9

7          CALL OPEN(1,"OUT5",2,IER)
           IF(IER.NE.1)TYPE"OPEN ERROR",IER

9          CALL RDBLK(1,0,IDATA,ISTOP,IER)
           IF(IER.NE.1)TYPE"RDBLK ERROR",IER

           CALL CLOSE(1,IER)
           IF(IER.NE.1)TYPE"CLOSE ERROR",IER

           RETURN
           END
```

```
            DO 182 I=1,10
182         CALL VTYPE(REJ(Z,I),WORD)
            GO TO 79
174         W=W+U
            V=W-1
            GO TO 171

179       · TOT(Y)=TOT(Y)/X
            Y=Y+1
            X=1
            FLAG=1
            V=1
            W=1·
            GO TO 171

1000        RETURN
            END         .
```

```
          S=1
          GO TO 172

C         CONTINUOUS RECOGNITION

177       V=1
          W=1
          X=1
          Y=1
          FLAG=0

171       DO 176 I=1,10
176       TWORD(I)=0

          TEMP3=9.0E60

          DO 173 I=1,10
          TWORD(I)=WRD(V)
          V=V+1
          IF(I.LE.2)GO TO 173

          CALL FINDWORD(PHON,TWORD,VOCAB,DIS,TEMP,L)
          IF(Y.LE.1)GO TO 180

          IF((L.EQ.REJ((Y-1),(Y-1))).AND.(FLAG.EQ.1))GO TO 186

180       IF(TEMP.GE.TEMP3)GO TO 173

          TEMP3=TEMP
          T=L
          U=I
          GO TO 173
186       FLAG=0
173       CONTINUE

          REJ(Y,X)=T
          TOT(Y)=TOT(Y)+TEMP3
          X=X+1
          IF((W+U).LT.S)GO TO 174
          IF(Y.LE.4)GO TO 179

          TOT(Y)=TOT(Y)/X
          S=1
          TEMP3=9.0E60

          DO 181 I=1,Y
          IF(TOT(I).GE.TEMP3)GO TO 181
          TEMP3=TOT(I)
          Z=I
181       CONTINUE
```

```
C*********************************************************************
C
C         Title:   RECOG.FR
C         Author:  Capt Ajmal Hussain
C                  Modified by 1Lt Kathy Dixon
C         Date:    Nov 84
C
C         Function:
C             Performs continuous speech recognition.  Prints
C             recognized string on video screen and line printer.
C
C*********************************************************************

          SUBROUTINE RECOG(PHON,VOCAB,WORD,DIS,LIB,J)

          INTEGER LIB(250),VOCAB(10,70),J,K,I,L,M,N,P,Q,R,S,T
          INTEGER U,V,W,X,Y,Z,TWORD(10),WRD(250),REJ(10,10)
          INTEGER FLAG,WORD(7,70),PHON(1130)
          REAL   DIS(2432),TOT(5)
          DOUBLE PRECISION REAL TEMP,TEMP1,TEMP3

          LEN1=2
          LEN2=9

          DO 188 I=1,70
          DO 188 K=1,10
188       REJ(K,I)=0

          DO 187 I=1,5
187       TOT(I)=0

          R=1
          S=1
172       DO 175 I=1,250
175       WRD(I)=0

79        IF(R.GE.(J+1))GO TO 1000
          IF(LIB(R).EQ.0)GO TO 170
          WRD(S)=LIB(R)
          R=R+1
          S=S+1
          GO TO 79
170       IF(S.GT.LEN1)GO TO 178
          R=R+1
          GO TO 79

178       IF(S.GT.LEN2)GO TO 177

          CALL FINDWORD(PHON,WRD,VOCAB,DIS,TEMP,L)

          CALL VTYPE(L,WORD)
```

```
C***********************************************************
C
C          Title:   PRINTREP.FR
C          Aithor:  1Lt Kathy R. Dixon
C          Date:    Nov 84
C
C          Function:
C             Prints table of vocabulary words and associated
C             phoneme representaition
C
C***********************************************************

          SUBROUTINE PRINTREP(WORD,VOCAB,NOWORDS)

          INTEGER WORD(7,70),VOCAB(10,70),NOWORDS

          WRITE(12,100)(WORD(1,I),(VOCAB(J,I),J=1,10),
     *    I=1,NOWORDS)
100       FORMAT(2X,S14,10X,10I4)

          RETURN
          END
```

102

```
        K=820
        DO 502 I=41,DIS(2416)
        WRITE(12,58)
        WRITE(12,54)I
        DO 503 J=1,40
        WRITE(12,54)(INT((DIS(((I*(I-1))/2)+J))))
503     CONTINUE
502     CONTINUE
        WRITE(12,59)
        WRITE(12,51)IOP
        WRITE(12,58)
        WRITE(12,58)
        WRITE(12,55)
        DO 505 I=41,DIS(2416)
        WRITE(12,54)I
505     CONTINUE
        DO 506 I=41,DIS(2416)
        WRITE(12,58)
        WRITE(12,54)I
        DO 507 J=41,DIS(2416)
        IF(J.GT.I)GO TO 507
        WRITE(12,54)(INT((DIS(((I*(I-1))/2)+J))))
507     CONTINUE
506     CONTINUE

500     RETURN
        END
```

```
C**********************************************************************
C
C          Title:   PRINTDIS.FR
C          Author:  Capt. Ajmal Hussain
C                   Modified by 1Lt Kathy Dixon
C          Date:    Nov 84
C
C          Function:
C              Prints lower triangular distance matrix.
C
C**********************************************************************

          SUBROUTINE PRINTDIS(DIS)

          REAL DIS(2432)

          IOP=1
          L=DIS(2416)
          WRITE(12,51)IOP
51        FORMAT(50X,"MAT",I2)
          WRITE(12,58)
          IF(DIS(2416).LE.40)GO TO 52
          L=40
          WRITE(12,55)
55        FORMAT("+",3X,Z)
52        DO 53 I=1,L
          WRITE(12,54) I
53        CONTINUE
54        FORMAT("+",I3,Z)
          K=1
          DO 56 I=1,L
          WRITE(12,58)
          WRITE(12,54) I
          DO 57 J=1,L
          IF(J.GT.I)GO TO 57
          WRITE(12,54)(INT(DIS(K)))
          K=K+1
57        CONTINUE
56        CONTINUE
58        FORMAT(1X)
          IF(DIS(2416).LE.40)GO TO 500

          WRITE(12,59)
59        FORMAT("1")
          WRITE(12,51)IOP
          WRITE(12,58)
          WRITE(12,58)
          WRITE(12,55)
          DO 501 I=1,40
          WRITE(12,54)I
501       CONTINUE
```

```
C***********************************************************
C
C          Title:   NORMALIZE.FR
C          Author:  Capt. Ajmal Hussain
C                   Modified by Kathy Dixon
C          Date:    Nov 84
C
C          Function:
C                   Normalizes sixteen dimensional vectors
C                   of digitized speech.
C
C***********************************************************

          SUBROUTINE NORMALIZE(IDATA2,IENERGY)

          COMMON/IBUFF/IDATA(8192)

          INTEGER IDATA2
          REAL IENERGY(500)
          DOUBLE PRECISION REAL TEMP

          TEMP=0
          K=1
          J=1
          L=1

          DO 5 I=1,(IDATA2/16)
          TEMP=0

          DO 2 J=K,(K+15)
          TEMP=TEMP+FLOAT(IDATA(J))**2      ;energy calculation
2         CONTINUE

          TEMP=(SQRT(TEMP)/32000)
          IENERGY(I)=ABS(TEMP)              ;store energy values
          DO 4 J=K,(K+15)
4         IDATA(J)=FLOAT(IDATA(J))/TEMP        ;energy normalize
                                               ;vectors
5         K=K+16

          RETURN
          END
```

99

```
C********************************************************************
C
C       Title: NewScr
C       Author: Lt Allen
C       Date: Dec 82
C
C       Function:
C          This routine erases the screen by typing 24
C          blank lines.
C
C       Compile command:
C          FORTRAN NEWSCR
C
C********************************************************************

        SUBROUTINE NEWSCR

        DO 10 I=1,24
        TYPE
 10     CONTINUE

        RETURN
        END

C********************************************************************
```

```
C****************************************************************
C
C          Title:   SPEECH.FR
C          Author:  1Lt Kathy R. Dixon
C          Date:    Nov 84
C
C          Function:
C             Calls routines to recognize continuous speech.
C
C****************************************************************

          SUBROUTINE SPEECH(ENRGY,PHON,DIS,VOCAB,WORD)

          COMMON/IBUFF/IDATA(4096)

          INTEGER PHON(1130),VOCAB(10,70),LIB(250),SOUND(250)
          INTEGER ISTOP,IFILE,IDATA2,WORD(7,70),ISWITCH,J
          REAL DIS(2432),IENERGY(500),ENRGY

          IDATA2=4000                        ;conversion value
          ISTOP=16                         ;last block to read
          IFILE=5                              ;file to read
          ISWITCH=1                 ;compress input string once

          CALL REDBUF(IFILE,ISTOP)                  ;read data

          CALL NORMALIZE(IDATA2,IENERGY)   ;normalize vectors

          CALL EXTRACT(IDATA2,PHON,IENERGY,ENRGY,LIB)
                                            ;extract phonemes

          CALL CMPRS(IDATA2,PHON,DIS,LIB,ISWITCH,SOUND,J)
                                           ;compress phonemes

          CALL RECOG(PHON,VOCAB,WORD,DIS,SOUND,J)
                                          ;determine words in
                                             ;input string
          RETURN                            ;print to screen
          END
```

```
C*********************************************************************
C
C          Title:   TEMPLATE.FR
C          Author:  1Lt Kathy Dixon
C          Date:    Nov 84
C
C          Function:
C              Performs necessary calls to support routines
C              to produce a phoneme template from 500 vectors
C              of input speech.  The template is stored in the
C              sequentially in the array, PHON.
C
C*********************************************************************

          SUBROUTINE TEMPLATE(ENRGY)

          INTEGER PHON(1130),NOVECT,IDATA2
          INTEGER WEIGHT(500),IWGHT(500),IWGHTT(500)
          REAL IENERGY(500),ENRGY

          COMMON/IBUFF/IDATA(8192)

          NOVECT=500
          IDATA2=8000                      ;conversion count
          ISTOP=32                         ;no. blocks to read
          IFILE=10                         ;file to read
          ENRGY=0

          CALL REDBUF(IFILE,ISTOP)         ;read data into buffer

          CALL NORMALIZE(IDATA2,IENERGY)   ;normalize vectors


C         WRITE(12,347)(IENERGY(I),I=1,500) ; option to write
347       FORMAT(10G11.5)                   ; energy values to line
                                            ;printer
          DO 88 I=1,500
88        ENRGY=ENRGY+IENERGY(I)

          ENRGY=(ENRGY/500)

          CALL LOWENERGY(IDATA2,IENERGY,ENRGY,WEIGHT)
                                            ;low energy vectors

          TYPE"Calculating template"

1         CALL CMPRS1(IDATA2,WEIGHT,IWGHTT) ;delete vectors

          DO 99 I=1,500
99        WEIGHT(I)=IWGHTT(I)
          NOVECT=IDATA(IDATA2+1)
```

113

```
          TYPE"Number of remaining vectors   ",NOVECT
          IF(NOVECT.LT.70)GO TO 2

          CALL COMPARE(IDATA2,WEIGHT,IWGHT)
                                         ;compare and average
          DO 44 I=1,500
44        WEIGHT(I)=IWGHT(I)
          GO TO 1                              ;nearest vectors

2         DO 5 I=1,1130               ;zero phoneme array
5         PHON(I)=0

          DO 3 I=1,1120             ;store phoneme template
3         PHON(I)=IDATA(I)

          PHON(1121)=NOVECT        ;store no. of phonemes

          CALL WRTPHON(PHON)

          RETURN
          END
```

```
C*****************************************************************
C                                                               *
C       Title:    TRAIN.FR                                       *
C       Author:   1Lt Kathy R. Dixon                            *
C       Date:     Jul 84                                        *
C                                                               *
C       Function:                                               *
C           This routine drives the system for training        *
C           and speech recognition for a specified user.       *
C                                                               *
C*****************************************************************


        INTEGER PHON(1130),SOUND(250),VOCAB(10,70)
        INTEGER WORD(7,70)
        REAL DIS(2432),ENRGY

1       CALL NEWSCR                                 ;erase screen

        TYPE"<CR>
  *     Welcome to the AFIT Speech Recognition Project.<CR>
  *     <CR>
  *     Using AFIT studies in speech research, a machine<CR>
  *     was developed to recognize continuous speech.<CR>
  *     Though training is required, the ultimate goal <CR>
  *     of this work is to develop a machine which is<CR>
  *     speaker independent.<CR>
  *     <CR>
  *     Continue? [Y]"

        CALL GCHAR(ICHAR,IER)
        IF(ICHAR.EQ.78)GO TO 111

        CALL NEWSCR

        TYPE"<CR>
  *     Press CR, then say the following<CR>
  *     phrases into the microphone after<CR>
  *     prompted by the word, start.<CR>
  *     Only 10 seconds of speech will<CR>
  *     be accepted.<CR>
  *     <CR>
  *     <CR>
  *     CHANGE FREQUENCY TO THREE FIVE SEVEN<CR>
  *     LOCK-ON TARGET AT TWO THOUSAND MARK<CR>
  *     ARM STATION ALFA BRAVO CHARLIE FOXTROT<CR>
  *     MAP AIR-TO-SURFACE MISSLE THREAT"

        CALL GCHAR(ICHAR,IER)

        CALL SWAP("ATOD10.SV",IER)                  ;swap for A/D
```

114

```
        IF(IER.NE.1)TYPE"SWAP error ",IER        ;10 sec speech

        CALL NEWSCR

        CALL TEMPLATE(ENRGY)              ;create phoneme template

        CALL NEWSCR

        TYPE"Calculating distance matrix"

        CALL GCHAR(ICHAR,IER)

        CALL REDPHON(PHON)

        CALL DISTANCE(PHON)               ;create distance matrix

        CALL REDDIS(DIS)

        CALL NEWSCR

        TYPE"Printing distance matrix"

        CALL GCHAR(ICHAR,IER)

        CALL PRINTDIS(DIS)                ;print distance matrix
34      CALL NEWSCR

        DO 15 I=1,70
        DO 15 J=1,10
15      VOCAB(J,I)=0                       ;zero VOCAB matrix

        NOWORDS=70                         ;no. vocabulary words

        CALL REDWRDS(NOWORDS,WORD)             ;read vocabulary
                                               ;from file
        TYPE"<CR>
    *   Press CR, a word will appear<CR>
    *   on the screen.  Press CR again<CR>
    *   then say the word.  Repeat. "

        CALL GCHAR(ICHAR,IER)

        DO 99 II=1,NOWORDS

        CALL NEWSCR

        WRITE(10,100) WORD(1,II)          ;write word to screen
100     FORMAT(2X,S14)

        CALL SWAP("ATOD2.SV",IER)                 ;swap for A/D
```

```
        IF(IER.NE.1)TYPE "SWAP error",IER          ;5 sec speech

        CALL REP(ENRGY,PHON,DIS,SOUND)

        DO 63 K=1,10
63      VOCAB(K,II)=SOUND(K)                        ;store word rep
99      CONTINUE

        CALL NEWSCR

        TYPE"Printing word phoneme representations"

        CALL WRTREP(VOCAB,NOWORDS)

        CALL PRINTREP(WORD,VOCAB,NOWORDS)           ;print word rep

        CALL REDREP(NOWORDS,VOCAB)

        CALL NEWSCR


C*****************************************************************


        TYPE"<CR>
    *       Training is complete.  Testing <CR>
    *       can now be done. An asterisk will<CR>
    *       appear on the screen.  Press CR,  <CR>
    *       then say the test word. The machine's<CR>
    *       guess will then be typed on the screen.<CR>
    *       <CR>
    *       Continue? [Y]"

        CALL GCHAR(ICHAR,IER)
        IF(ICHAR.EQ.78)GO TO 111

77      CALL NEWSCR

        TYPE"*"                                     ;request speech input


        CALL GCHAR(ICHAR,IER)

        CALL SWAP("ATOD5.SV",IER)                   ;swap for A/D
        IF(IER.NE.1)TYPE"SWAP ERROR",IER            ;5 sec speech


        CALL SPEECH(ENRGY,PHON,DIS,VOCAB,WORD)      ;speech
                                                    ;recognition
```

116

```
        TYPE"<CR>
*     Continue? [Y] "

      CALL GCHAR(ICHAR,IER)
      IF(ICHAR.EQ.78) GO TO 73

      GO TO 77

73    CALL NEWSCR

      TYPE"<CR>
*     This concludes the session.   <CR>
*     End session? [Y] "

      CALL GCHAR(ICHAR,IER)
      IF(ICHAR.EQ.78) GO TO 111
      GO TO 112

111   TYPE"<CR>
*     Start again? [N] "

      CALL GCHAR(ICHAR,IER)
      IF(ICHAR.EQ.89) GO TO 1

112   CALL EXIT
      END
```

```
C****************************************************************
C
C     Title: VTYPE.FR
C     Author: 1Lt Kathy Dixon
C             Modelled after Hussain's WTYPE
C     Date:   Nov 84
C
C     Function:
C        This routine displays the word specified on the
C        H19 terminal in video on a single line.
C
C
C****************************************************************

      SUBROUTINE VTYPE(L,WORD)

      INTEGER L,WORD(7,70)

      IF(L.EQ.0) GO TO 10              ;skip zero numbered words

      L=(L-1)/10+1
      WRITE(12,15)L
      WRITE(12,14)WORD(1,L)

      WRITE(10,11)WORD(1,L)                    ;display word

11        FORMAT(S14)
14        FORMAT("+",S14,Z)
15        FORMAT(2X,I4)

10    RETURN
      END

C****************************************************************
```

118

```
C*******************************************************************
C
C        Title:   WRTDIS.FR
C        Author:  Kathy Dixon
C        Date:    Nov 84
C
C        Function:  Writes Distance matrix to a disk file.
C
C*******************************************************************

         SUBROUTINE WRTDIS(DIS)

         REAL DIS(2432)

         CALL CFILW("DIST",2,IER)
         IF(IER.NE.1)TYPE"CFILW ERROR",IER
         CALL OPEN(1,"DIST",2,IER)
         IF(IER.NE.1)TYPE"OPEN ERROR",IER
         WRITE(1,100)(DIS(I),I=1,2432)
100      FORMAT(G11.5)
         CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE"CLOSE ERROR",IER

         RETURN
         END
```

```
C******************************************************************
C
C         Title:   WRTPHON.FR
C         Author:  Kathy Dixon
C         Date:    Nov 84
C
C         Function:  Writes 70 16-dimensional vectors to
C                    a file, PHONE
C
C******************************************************************

          SUBROUTINE WRTPHON(PHON)

          INTEGER PHON(1130)

          CALL CFILW("PHONE",2,IER)
          IF(IER.NE.1)TYPE"WRTPHON CFILW ERROR",IER
          CALL OPEN(1,"PHONE",2,IER)
          IF(IER.NE.1)TYPE"WRTPHON ERROR",IER
          WRITE(1,100)(PHON(I),I=1,1130)
100       FORMAT(I6)
          CALL CLOSE(1,IER)
          IF(IER.NE.1)TYPE"CLOSE ERROR",IER

          RETURN
          END
```

```
C*********************************************************
C
C       Title:  WRTREP.FR
C       Author: Kathy Dixon
C       Date:   Nov 84
C
C       Function:  Writes word phonemes to a file, VOCABUL.
C
C*********************************************************

        SUBROUTINE WRTREP(VOCAB,NOWORDS)

        INTEGER VOCAB(10,70),NOWORDS

        CALL CFILW("VOCABUL",2,IER)
        IF(IER.NE.1)TYPE"CFILW ERROR",IER
        CALL OPEN(1,"VOCABUL",2,IER)
        IF(IER.NE.1)TYPE"OPEN ERROR",IER
        WRITE(1,100)((VOCAB(I,J),I=1,10),J=1,NOWORDS)
100     FORMAT(I3)
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE ERROR",IER

        RETURN
        END
```

```
C*************************************************************
C
C       Title:  VOCAB.FR
C       Author: 1Lt Kathy Dixon
C       Date:   Nov 84
C
C       Function:
C           Creates a file of English words.
C
C       Command Line:
C           RLDR VOCAB NEWSCR ?FLIB?
C
C*************************************************************

        INTEGER W(7,70),L

        ACCEPT"NUMBER OF VOCABULARY WORDS ",L
        TYPE"       *    *     *     *"
        DO 200 I=1,L
        ACCEPT"WORD "
        READ(11,100)W(1,I)
100     FORMAT(S14)

200     CONTINUE

        ACCEPT"PRINT VOCABULARY [Y]"
        CALL GCHAR(ICHAR,IER)
        IF(ICHAR.EQ.78)GO TO 111

        CALL NEWSCR

        WRITE(10,150)(W(1,I),I=1,L)
        WRITE(12,150)(W(1,I),I=1,L)
150     FORMAT(2X,5S14)

        CALL CFILW("WORDS",2,IER)
        IF(IER.NE.1)TYPE"CFILW ERROR",IER

        CALL OPEN(1,"WORDS",2,IER)
        IF(IER.NE.1)TYPE"OPEN ERROR",IER
        WRITE(1,100)(W(1,I),I=1,70)
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE ERROR",IER

111     CALL EXIT
        END
```

VITA

Kathy Renee Dixon was born on 27 January 1959 in Roswell, New Mexico. She graduated form high school in Orlando, Florida in 1977 and attended the University of Central Florida, Orlando, Florida from which she received the degree of Bachelor of Science in Engineering in April 1982. She entered the Air Force on active duty in May 1982 and received her commission from Officer Training School in August 1982. She served as a Project Engineer in the Signal Processing Laboratory, Air Force Institute of Technology until entering the School of Engineering in June 1983.

Permanent address: 520 Sheppard Road
                       Orlando, Florida 32820

AD-A151936

REPORT DOCUMENTATION PAGE

URITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
~ASSIFIED	

LASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
	Approved for public release,
CATION/DOWNGRADING SCHEDULE	distribution unlimited

ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
'ENG/84D-26	

RFORMING ORGANIZATION	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
of Engineering	AFIT/ENG	

ity, State and ZIP Code)	7b. ADDRESS (City, State and ZIP Code)
ce Institute of Technology	
Patterson AFB, Ohio 45433	

JNDING/SPONSORING ~ION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

'ity, State and ZIP Code)	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

de Security Classification)
J9

AUTHOR(S)

, Dixon, B.S.E., 1st Lt, USAF

EPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
is	FROM _____ TO _____	1984 December	135

ITARY NOTATION

COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
ROUP	SUB. GR.	Recognition, Speech, Phonemes, Speech Recognition,
02		Continuous Speech Recognition

(Continue on reverse if necessary and identify by block number)

IMPLEMENTATION OF A REAL-TIME,
INTERACTIVE, CONTINUOUS SPEECH
RECOGNITION SYSTEM

Chairman: Dr. Matthew Kabrisky, Professor of Electrical Engineering,
       Air Force Institute of Technology

ION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
/UNLIMITED ☒ SAME AS RPT. ☐ OTIC USERS ☐	UNCLASSIFIED

RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
:thew Kabrisky, Professor, AFIT	513-255-5276	AFIT/ENC

A speech recognition system was designed and implemented to recognize continuous speech in a real-time environment (after training). Several techniques were incorporated to characterize phonemes as vectors in space. Through the use of distance rules, it was possible to characterize words by a phoneme representation. This approach to speech recognition offers several possibilities for future investigationssuch as varying Minkowski distances and applying clustering techniques. The algorithm which was developed was modularized on a hierarchical basis and was user friendly.

END

FILMED

4-85

DTIC